Self-Organizing Network (SON) Functionalities for Mobility Management in WiFi Networks

by

Lina Hao

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Network Engineering.

Victoria University of Wellington
2019

Abstract

WiFi networks based on the IEEE 802.11 standard are widely used indoors or outdoors as simple and cost-effective wireless technology. However, the data connection is significantly disrupted when mobile stations (STAs) switch between access points (APs). Furthermore, high packet loss occurs during the switching period. Therefore, mobility is a critical issue that needs to be solved in WiFi networks.

In cellular networks, handover is used to keep ongoing data transfer when network clients switch between base stations. However, the handover algorithm is not supported in the 802.11 standard for WiFi networks. Self-Organizing Network (SON) functionality enables seamless handover in cellular networks, improving network performance. However, the SON functionality has not been fully researched in WiFi networks, especially for mobility management.

Motivated by the SON functionalities, a SON approach is proposed to automatically optimize the handover algorithms for WiFi networks. This approach focuses on the SON functionalities including self-configuration, self-optimization and self-healing using machine learning techniques to develop new algorithms for WiFi mobility management. The overall goal of this thesis is to optimize handover performance as well as enhance the network's capabilities.

Acknowledgements

I would like to thank my primary supervisor Dr Bryan Ng, who provide endless support in every aspect during my PhD life. Your encouragement made me have confidence in my research. I also would like to thank my secondary supervisor professor Winston Seah, your valuable advice and comments on my research help me to widen my research from various aspects. I appreciate my parents' support and encouragement when I felt hopeless in my PhD journey. A special thanks for my son, Haoyuan Gu, who always accompany me when I work on research in the evening and weekend. Besides, I would like to thank my beloved husband, Cheng Gu, who provides spiritual and financial support to my PhD study.

Publications

- Lina Hao, Bryan Ng and Ying Qu, *"Dynamic optimization of neighbor list to reduce changeover latency for wi-fi networks"*. Proceedings of 2017 International Conference on Telecommunications and Communication Engineering (ICTCE), October 22-24, 2017, Osaka, Japan.

- Lina Hao, Bryan Ng, *"Using Genetic Algorithms based on Neighbor List Mechanism to Reduce Handover Latency for IEEE 802.11 WLAN"*. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), July 15-19, 2018, Kyoto, Japan.

- Lina Hao, Bryan Ng and Ying Qu, *"Self-Optimizing Scanning Parameters for Seamless Handover in IEEE 802.11 WLAN"*, Proceedings of the 43nd IEEE Conference on Local Computer Networks (LCN), October 1-4, 2018, Chicago, USA.

- Lina Hao, Bryan Ng, *"Self-healing solutions for Wi-Fi networks to provide seamless handover"*, Proceedings of the 16th IFIP/IEEE International Symposium on Integrated Network Management (IM) , April 8-12, 2019, Washington DC, USA.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

IEEE 802.11-based wireless local area networks (WLANs) have many advantages including low cost, easy deployment and high bandwidth capabilities [1]. For these reasons, they are used throughout campuses or enterprises. The 802.11 standard for WiFi was designed for an indoor network solution where the network clients are mostly stationary. However, WiFi has been used where clients move, both indoors and outdoors. As the mobile stations (STAs) move between access points (APs), there is a significant disruption to ongoing data connections. Therefore, mobility is becoming a critical issue in meeting real-time network requirements.

Handover can be used to keep the data connection active when STAs move from one AP to another in wireless networks. However, handover is not explicitly supported in the 802.11 standard for WiFi networks [2]. The standard 802.11 only supports the STAs switching between two APs when the STAs move out of range of a serving AP. During the switching period, the STAs break their association with the serving AP and the STAs are potentially unable to communicate any type of data traffic within this time. Therefore, handover algorithms need to be designed for mobility management in WiFi networks.

Although some research has been done to improve throughput and reduce packet loss during handovers for WiFi networks, this research only considers one aspect to improve handover performance such as reducing the handover delay [3–5] or optimizing the handover related parameters [6–8]. Most research optimizes handover algorithms based on normal network conditions. However, outages in WiFi networks need to be considered in handover algorithms to meet the dynamic network requirements. Thus, seamless handover algorithms without any considerable degradation of the quality of service required by applications [9] need to be investigated.

1.1 Motivation

Self-Organizing Network (SON) technology has been used in cellular networks as a dynamic solution for handover issues including handover failures and handover delay [10–13]. The SON technology comprises three functionalities: self-configuration, self-optimization and self-healing. One function of self-configuration is to automatically manage the neighbor cells relations using a Neighbor Cell List (NCL). Fast handover is achieved based on the neighbor information provided by NCL. With the function of self-optimization, the handover parameters are dynamically updated based on the real-time network conditions, thus both the handover delay and the handover failure rate is reduced. Finally, self-healing automatically detects errors and performs recovery without downgrading the network service. Therefore, all three of these SON functions combined can overcome the handover issues [14].

Motivated by the SON technologies applied in cellular networks, a new SON approach needs to be developed to fit the gap of limited handover management in WiFi networks. With the SON approach used in WiFi networks, the data disruption time and packet loss rate may be reduced to keep ongoing data connections active when a STA switches between APs. SON provides an approach for introducing mobility management into WiFi networks.

1.2 Research Goals

The goal of this research is to solve mobile quality of service (QoS) issues using SON functionality including configuration, optimization and network healing in WiFi networks.

- *Objective 1: To reduce handover delay using dynamic self-configuring neighbor lists*

  This objective is to investigate updating strategies of neighbor lists based on the IEEE 802.11k standard for radio management. The IEEE 802.11k standard provides mechanisms for gathering data including Received Signal Strength Indicator (RSSI), Signal to Noise Ratio (SNR), noise histogram request/report and BSS average delay. The self-configuring neighbor list will be updated based on these capabilities of the radio measurement reports and information. This method can reduce the number of channels scanned and handover decision time in turn to reduce the handover delay.

- *Objective 2: To optimize the scanning parameters using self-optimizing algorithms*

  This objective is to investigate an adaptive scanning method to automatically adjust scanning parameters based on the real-time network conditions. The values of the scanning parameters are not specified in the 802.11 standard, instead these are fixed values pre-defined by vendors. These fixed values cannot satisfy the real-time network environment and cause high scanning delay. A self-optimizing scanning method is necessary to reduce the unnecessary waiting time of probe responses of non-adjacent APs or faulty APs.

- *Objective 3: To automatically diagnose network performance problems and compensate for network degradation using a self-healing method*

This objective is to investigate an automated method to monitor and detect the APs with degraded performance. Machine learning algorithms will be investigated to find out how to detect an abnormal AP. A corrective action will be used for network compensation when an AP becomes abnormal.

1.3 Research Methodology

The main objective of this thesis is to solve the data disruption issues when STAs switch between APs in WiFi networks. To achieve this objective, both mathematical analysis and simulations have been used to study network performance.

Mathematical Analysis is used because the mathematical formulation provides a clear relationship between the input and output. In this study, the results of the output cannot be easily obtained in dynamic network environments. Therefore, machine learning algorithms and Genetic Algorithms (GA) have been used to obtain the solutions of optimal output after the problem is formulated. In Chapter 3, neighbor list updating is formulated as a ranking problem. Three machine learning models are applied to solve the ranking problem and study network performance. In Chapter 4, GA is used for solving a combinatorial optimization problem to dynamically adjust scanning parameters. In Chapter 5, detecting a faulty AP is constructed as a binary classification problem. Five machine learning algorithms are applied for faulty AP detection and used for network performance evaluation.

However, machine-learning-based approaches need network data to predict the results. Some network data can be obtained by some vendors but these data are not public for academic research. Although free data is available from network operators such as Google geolocation API, open signal or websites such as ′′data.world′′, these data cannot be applied directly to IEEE 802.11k standard for solving the handover issues. Therefore, the network data needs to be collected by simulation or real measurement in different network scenarios.

Using testbed is good to get the real measurement data. However, it is difficult and expensive to set up testbed experiments with a large number of nodes and multiple network topologies on campus. Simulation can easily construct different network scenarios and collect network data. Therefore, simulation is used to evaluate network performance in this thesis. Simulations are also used for validation of the proposed approaches by comparing simulation results with conventional handover methods and related studies in the literature.

1.4 Research Contributions

This thesis contributes to a SON approach including a self-configuring neighbor list, a self-optimizing scanning algorithm and a self-healing system for mobility management in WiFi networks. Specifically, a neighbor list mechanism is designed for AP neighborhood management. With a neighbor list, the scanning delay is reduced because the neighbor list reduces the number of channels and APs scanned. The scanning parameters have been optimized by using Genetic Algorithms (GA) and adaptive timers. The overall network performance has been improved by these self-optimizing algorithms. Moreover, a self-healing system is implemented to detect abnormal APs. Degraded network performance has been compensated for due to the use of self-healing approach.

- **A self-configuring neighbor list mechanism**
  This neighbor list mechanism will automatically establish up-to-date neighbor lists to reduce handover delay. The neighbor list is updated periodically with radio parameters such as SNR, load and delay as STAs start a new scan. With the reduction of channels and APs using the neighbor lists, the scanning delay is reduced. Moreover, the APs

on the neighbor list are sorted by Quick-sort or machine learning algorithms. During the handover, the STA will re-associate with the highest priority AP in the neighbor list.

When using the machine learning based neighbor list mechanism, the APs are sorted based on the AP score. The AP score considers several network predictors including RSSI, SNR, packet delay, packet loss rate, data rate, throughput and AP load. The handover decision is made based on the AP score instead of only one predictor and is updated to meet dynamic network requirements. Therefore, the total handover delay is reduced by using dynamic handover decisions.

The contribution of this approach not only reduces the handover delay but also improves the handover performance.

- **A self-optimizing scanning algorithm**
  Some researchers optimize the handover delay by tuning the threshold of predictors including RSSI, SNR or frame re-transmissions. Although these solutions have reduced handover delay, the solutions are based on the fixed probe timers *MinChannelTime* and *MaxChannelTime*. The timer *MinChannelTime* is the minimum waiting time a STA undergoes before considering that the channel is empty while the timer *MaxChannelTime* is the maximum waiting time after a probe response has been successfully received. Although the probe timers (*MinChannelTime* and *MaxChannelTime*) or the number of channels scanned are optimized in some research to reduce the scanning delay, all related scanning parameters need to be optimized. The related scanning parameters include the probe timers, the probe request interval, channel switching time, the number of scanned channels and the channel sequence.

  In this thesis, all the related scanning parameters have been optimized by GA which is a simple and efficient algorithm to find the global optimum of scanning parameters to meet the dynamic network requirements. These self-optimizing adaptive parameters are dynamically adjusted based on real-time network conditions to reduce MAC layer handover delay. Each STA is made aware of the scanning timers for each channel by a centralized controller. Therefore, the STA probes a set of channels and APs with adaptive timers instead of fixed values. The proposed algorithms reduce the number of scanned channels and probed access points and remove the unnecessary waiting time of probe responses of non-adjacent APs or faulty APs.

- **A self-healing system**
  A new self-healing solution is proposed to reduce the handover failure rate and improve network performance. The self-healing function includes self-detection and self-compensation. The self-detection function detects and locates faults promptly, accurately, and automatically by processing network statistics. The self-compensation function recovers or compensates for any breakdowns to consistent, high-quality communication.

  In this thesis, self-healing is used to monitor faulty APs and compensate for network degradation. When an AP with degraded performance is detected, the faulty AP will be added to the blacklist preventing STAs from associating with the faulty AP. At the same time, the current STAs that are associated with the faulty APs are forced to handover to neighbor APs to compensate for the network degradation. Machine learning algorithms are used for the self-detection of faulty APs. The neighbor list mechanism provides the neighbor APs′ information to the faulty AP in preparation for an impending handover. Self-healing reduces the handover failure rate and compensates for network degradation.

1.5 Organisation of Thesis

The remainder of this thesis is outlined as follows. Chapter 2 presents a comprehensive review of the background and related work in the field of WiFi handover. The next three chapters introduce WiFi SON functionalities including the neighbor list mechanism, self-optimizing scanning parameters and self-healing solutions, respectively. Finally, the content and outcome of this thesis are summarized in chapter 6. Several potential research paths are also suggested to benefit the future development of SON.

Chapter 2

Background and Related Work

This research investigates approaches to improve handover performance for IEEE 802.11 Wireless LAN networks (WLANs). Firstly, some key terms used in this research are given and the IEEE 802.11 handover background is introduced. Secondly, the handover issues and solutions are summarised based on related works. Thirdly, the Self-Organizing Networks (SON) functionalities used in cellular networks and the challenges of SON in WiFi networks are introduced. Finally, the methods of handover evaluation are discussed. This chapter summarizes the issues in WiFi handover and SON technologies that motivate this research.

2.1 Definitions of Key Terms

This section introduces some key terms that are present in an IEEE 802.11 network. These key terms are used in the whole thesis to facilitate the discussion.

- **IEEE 802.11 networks**: IEEE 802.11 is a set of media access control (MAC) and physical layer (PHY) protocols to implement wireless local area network (WLAN). The initial standard of 802.11 started in the late 1990s. The initial standard has been extended to include amendments related to the increase of throughput (IEEE 802.11a, b, g, n, ac, ad, ax), information reporting (IEEE 802.11k, u, v) or enhancing transitions (IEEE 802.11r, ai, aq) [15].

- **IEEE 802.11a**: This amendment was released in 1999. It provides data rates of 1.5 to 54 Mbps using 5 GHz band. It was designed for Orthogonal Frequency Division Multiplexing (OFDM) communication system. The modulation type includes Binary Phase Shift Keying (BPSK), Quadrature Phase Shift Keying (QPSK), 16-Quadrature amplitude modulation (QAM) and 64-QAM [16].

- **IEEE 802.11b**: This amendment was released in 1999. It provides data rates up to 11 Mbps using 2.4 GHz band. The modulation type used is complementary code keying (CCK) in 22 MHz wide channels [17].

- **IEEE 802.11g**: This amendment was released in 2003. It provides data rates of up to 54 Mbps using 2.4 GHz band. The modulation type used is CCK in 20 MHz wide channels [18].

- **IEEE 802.11k**: This amendment was released in 2007. It is used for radio resource management. It defines and exposes radio and network information to facilitate the management and maintenance of a mobile Wireless LAN [19].

- **IEEE 802.11r**: This amendment was released in 2008. It is used to permit continuous connectivity aboard wireless devices in motion, with fast and secure handovers from one base station to another managed in a seamless manner [20].

- **Station (STA)**: STA is a device can use the 802.11 protocol. Any device that contains an IEEE 802.11-conformant MAC and PHY interface to the wireless medium [15].

- **Access Point (AP)**: AP is network hardware to provide access to the wired network. If an AP is present, all communications between two STAs are done through the AP, even though both STAs were close enough to contact directly [15].

2.2 Standard 802.11 Handover

In the IEEE 802.11 standard, handover is the process of transferring ongoing calls or data sessions when a STA moves from one access point (AP) to another AP. In order to make a handover, the STA has to decide when to handover. However, according to the IEEE 802.11 standard [15], there is no specific handover algorithm to support such mobility management. IEEE 802.11 only supports a STA to change its association from one access point to another owing to poor quality link. Some researchers use predefined thresholds including Received Signal Strength Indicator (RSSI) or Signal to Noise Ratio (SNR) to trigger a handover [21, 22]. For example, when a STA moves away from its current AP and the RSSI is below the handover RSSI threshold, a handover procedure is triggered.

A basic 802.11 handover process includes three phases: scanning/probe, re-authentication and re-association. Scanning of channels for APs can be accomplished in two ways according to the IEEE 802.11 standard, passive and active [15].

In the passive mode, a STA selects an AP based on the information including Basic Service Set Identifier (BSSID) and the signal strength from beacon frames. The STA has to listen to the beacon frames from all APs on each channel. Therefore, the STA at least waits for one beacon interval about 100ms on each channel. The passive scanning delay T_{PS} is given by

$$T_{PS} = N \times BeaconInterval + N \times ChannelSwitchTime \tag{2.1}$$

where $BeaconInterval$ is the beacon interval, $ChannelSwitchingTime$ is the time that STA switches from one channel to another and N is the number of channels in the IEEE 802.11 spectrum.

In the active mode, the STA broadcasts probe requests and waits for probe responses from APs actively without listening to any beacon frames. Compared to passive scan, the active scan has less waiting time and is more suitable for realizing seamless handover in 802.11 WLANs [23]. Therefore, the active scan is considered to further reduce the handover delay in this thesis.

The active scanning delay T_{AS} is given by

$$\begin{cases} N \times MinChannelTime + N \times \delta \leq T_{AS}, \\ T_{AS} \leq N \times MaxChannelTime + N \times \delta \end{cases} \tag{2.2}$$

where δ is the channel switching time, N is the number of channels in the IEEE 802.11 spectrum, *MinChannelTime* and *MaxChannelTime* are the scanning timers. δ varies from 5 to 10ms specified by vendor implementation.

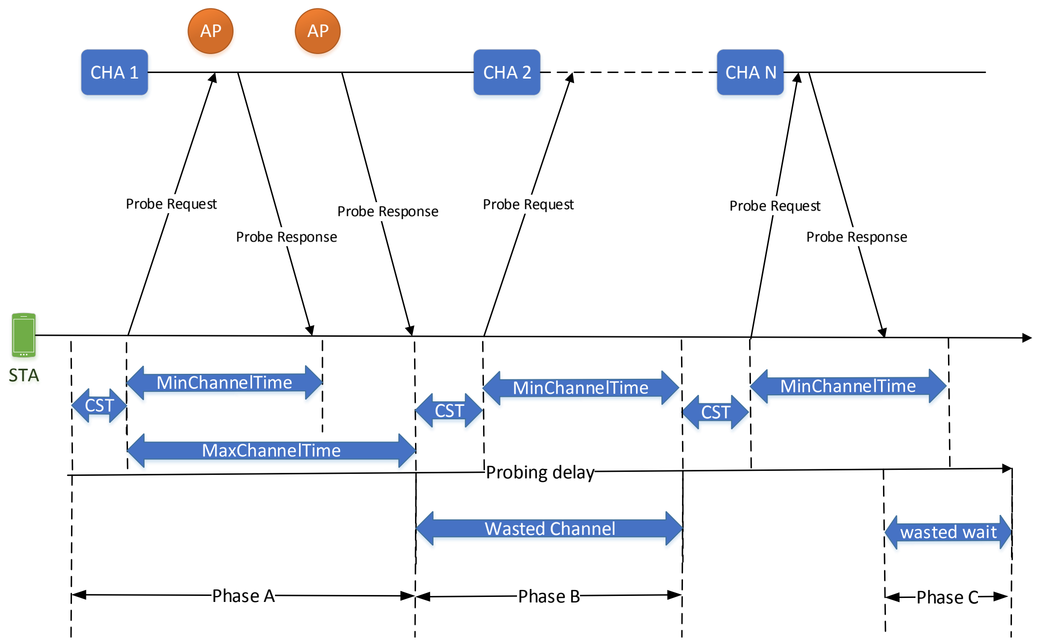


Figure 2.1: Active scanning process in 802.11 handover

The active scanning process based on the IEEE 802.11 standard [15] follows steps which are depicted in Figure 2.1. The handover procedures using active scanning are illustrated in Figure 2.2 and the details of each step are explained in the following paragraphs.

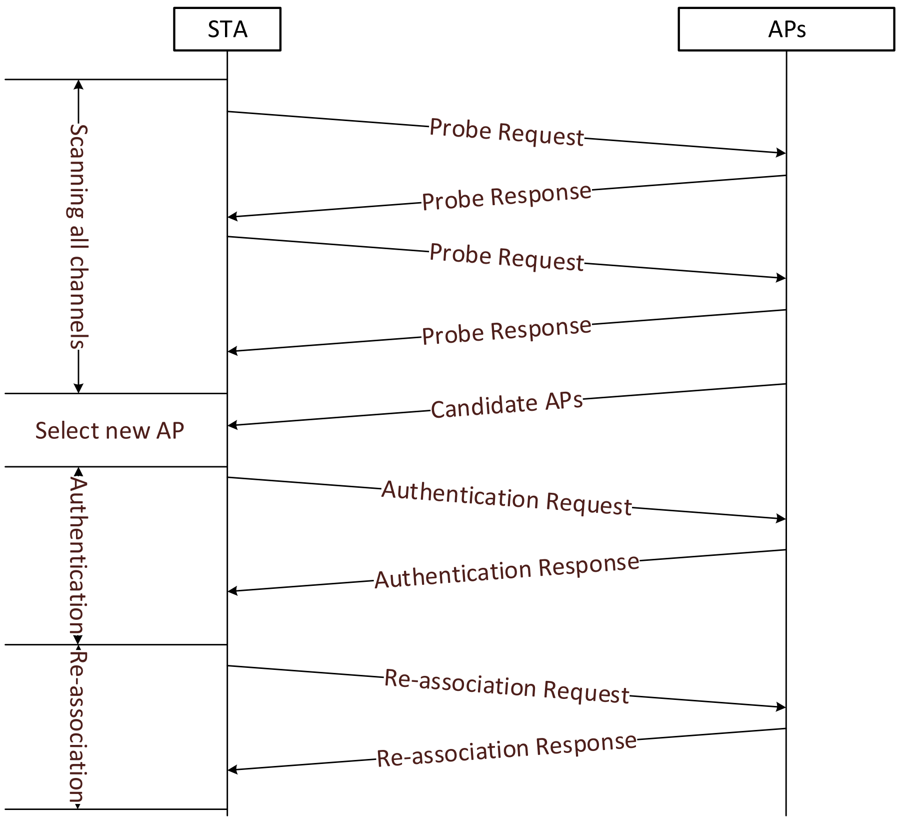


Figure 2.2: The standard 802.11 handover procedures using active scanning

Firstly, the STA in the scanning phase starts to probe by switching to channel 1 (CHA 1) and broadcasts a probe request frame to APs. The STA then waits for probe responses on that channel, if no response has been received within a time *MinChannelTime* (measured in seconds), the STA probes the next channel. If one or more responses are received within *MinChannelTime*, the probing STA continues accepting probe responses until

MaxChannelTime (Phase A in Figure 2.1).

Next, the STA moves to the next channel and repeats the above steps. The Channel Switching Time is termed as 'CST'. After STA scans all the channels, the information received from probe responses is processed and the STA can select the best AP to join next. Note that there is wasted time on channel 2 (CHA 2) when no AP exists (wasted channel) (Phase B Figure 2.1). Moreover, the STA keeps waiting for more probe responses on channel N (CHA N) even after receiving the last probe response, which is a wasted probe-wait (Phase C Figure 2.1). Thus, reducing the wasted waiting time on non-AP channels and optimizing the probe timers can reduce the scanning delay and in turn shorten the total handover delay.

After active scanning, the STA sends an authentication request to the selected AP. The selected AP will send an authentication response frame indicating whether to accept this STA or not.

If the STA is accepted by the selected AP, the re-association process is performed. The STA sends a re-association request to the selected AP and a re-association response containing acceptance or rejection of the STA will be sent by this AP.

2.3 Handover Issues and Past Works

During the period of handover from the old AP to the new AP, the STA breaks its association with the old AP and it is potentially unable to communicate any type of data traffic within this time. This break time is called handover delay. The total handover delay includes scanning delay, re-association delay and re-authentication delay [8]. In this thesis, active scanning is considered. Therefore, the handover delay can be calculated by Equation 2.3,

$$T_{handover} = T_{scanning} + T_{authentication} + T_{re-association} \tag{2.3}$$

The handover delay is about 200ms on average in an IEEE 802.11b networks using 11 channels and active scanning [6]. Though active scanning is much faster, it cannot satisfy the real-time applications whose delay requirement is no more than 50ms [24]. Therefore, many researchers have discussed the potential optimizations of the handover delay.

In the rest of this section, the existing solutions for how to reduce handover delay and improve handover performance will be discussed.

2.3.1 Reducing of the Number of Scanned Channels

The reduction of the number of channels needed to be scanned is one way to shorten the scanning delay. This solution reduces handover delay sharply, but the trade-off is that it requires information of APs prior to a handover. Therefore, some extra databases or caching mechanisms are usually needed to provide such information. The drawback is that keeping the information of APs up-to-date may pose challenges to the scalability of the network.

Some researchers collected the neighbor APs information before handover performance and aimed to reduce the number of channels scanned. The neighbor APs are the candidate APs for next handover. The total scanning delay can be reduced because only the channels of the neighbor APs are scanned. Thus, the total number of APs scanned is reduced. Mishra et al. [3] established neighborhood relationships based on Neighbor Graph (NG). NG is generated using the re-association request message from an STA containing BSSID. NG reduces not only the number of channels scanned but also the time wasted in scanning the empty channels without APs [25]. Thus, the handover delay is significantly reduced.

Another way to reduce the number of channels scanned is to use selective scanning. For selective scanning, S. Shin et al. [4] selected the scanned channels by a data structure called channel mask. STA only scan the selected channels from non-overlapping channels (1, 6 and 11). As the probability of a neighbor AP on the same channel of the current AP is small, the channel of the previously connected AP will not be scanned. Combined with the caching mechanism for storing information about surrounding APs in the previous handovers, the latency attains further reduction.

The Global Positioning System (GPS) is also applied for selecting scanning APs based on the location information provided by GPS receivers. The location information requested by a STA can be obtained from its GPS receiver, and the cooperation of APs is supported by a location server. Tseng et al. [5] introduced a location-based handover algorithm. If an AP is located within a certain range of movement direction of the STA, it will be included in a candidate list as a potential AP for handover. Whenever a handover occurs, only the channels on the candidate list will be scanned. Therefore, the scanning delay is reduced.

In summary, the way to reduce the number of channels scanned is to establish a list of candidate APs for the next handover. The AP list discussed above is not an up-to-date list based on the wireless network conditions. Moreover, the candidate APs are not sorted and updated periodically on the list. The sorted APs on the list will also reduce the probe waiting time of probe responses. Therefore, an up-to-date AP list based on networks conditions is highly needed.

2.3.2 Optimizing Scanning Parameters

The discussion above about how to reduce the number of channels scanned is based on the fixed scanning parameters. Optimizing the scanning parameters is another method to reduce the scanning delay. The scanning parameters *MinChannelTime* and *MaxChannelTime* have a big influence on scanning delay because the probe waiting timer depends on these two values. The probe waiting time is the time an STA waits on one particular channel after sending a probe request message. Thus, the STA waits on one channel for at least *MinChannelTime*. If any traffic or probe response message was received within *MinChannelTime*, the STA extends the probe waiting time to *MaxChannelTime*. If the values of these two parameters are too high, unnecessary waiting time is wasted. On the other hand, a STA may not have enough time to receive all probe responses from APs, if the values are too low. Therefore, there is a trade-off between the AP discovery rate and total handover delay. How to automatically optimize the scanning parameters has a significant effect on the successful handover rate.

The total probe delay t, for probing N channels, is defined as follows:

$$N \times MinChannelTime \leq t \leq N \times MaxChannelTime \tag{2.4}$$

As the values of *MinChannelTime* or *MaxChannelTime* are not defined in the IEEE 802.11 standard, some researchers search for the optimal values of them based on experiments. A.Mishra et al. recommended that the proper values of *MinChannelTime* and *MaxChannelTime* should be 6.5ms and 11ms, respectively [26] based on their extensive tests. The authors in [6] presented an extensive analysis of the scanning process by mean of simulation and real testbed. They observe that there is no single fixed optimal pair (*MinChannelTime* and *MaxChannelTime*) that always gives the best scanning performance in all deployment, which is consistent with earlier studies [27] [28].

A few researchers have studied how to dynamically optimize the scanning parameters such as *MinChannelTime* or *MaxChannelTime* to reduce the probe waiting time. However, the adaption algorithms are based on some assumptions. For example, the algorithm in [6] uses an adaptation method for adjusting the scanning timers in order to increase the discovery rate. This method assumes the discovery rate in the last scan is accurate. The scanning timers are increased when no AP is found and decreased when an AP is found in the last scan. The work in [6] departs from the trend of increasingly complex algorithms and argues for a simple yet practical solution. Also, R. Pazzi et al. [7] and A. Boukerche et al. [8] proposed scanning algorithms adapted to the scanning timers. The timers are dynamically adjusted based on the predicted channel information. However, the ideas in both [7] and [8] predict the target handover AP probability on each channel based on the assumption that the information collected from the last channel scan is accurate. This assumption is overly burdensome and requires significantly higher computational complexity because it maintains the state of the channel for each known AP.

In a recent paper [29], an intelligent module using Cultural Algorithm (CA) is introduced to get a sequence channel with optimized scanning timers. In this work, the optimization problem is formulated as double objective problems which aim to characterize the trade-off for a better discovery process by using the discovery rate (APs per time unit) versus the total latency for scanning. The optimal channel scanning sequence, the number of AP discovered and its correspondent scanning timers including *MinChannelTime* and *MaxChannelTime* are considered as the output of CA. However, the channel sequence is sorted according to the number of AP discovered. The number of AP discovered based on CA cannot reflect the real channel conditions in the wireless network. This work is based on the assumption that CA can adapt to the channel varying in different network topologies.

Overall, the scanning parameters are needed to dynamically update in the different network environment. Although some research works proposed adaptive scanning parameters based on some prior or prediction assumptions, these optimized scanning timers cannot adapt to the real-time network environment. Thus, the handover performance is only enhanced with some restrictions in a certain network environment.

2.3.3 Pre-scan Strategy

The handover delay also can be shorted when the scanning phase is executed before a handover is triggered. Such schemes are usually able to remove the scanning delay completely, but they increase system complexity and introduce overheads.

SyncScan [30] is one such protocol that can sharply reduce handover latency. APs are synchronized to send out beacons at fixed times so that a STA can switch to other channels at the same time to check the signal strength of a certain AP. Continuing this process, the STA will have complete knowledge of surrounding APs in advance and then will select the AP with the strongest signal, whenever a handover is required. SyncScan does add some complexities, as time synchronization is required, which can be achieved by NTP (Network Time Protocol) [31]. The main issue, however, is that high throughput cannot be achieved, because the working channel of the STA periodically changes all the time.

DeuceScan [32]implements a similar pre-scan scheme for vehicular environments. The STA probes all channels first and then it constructs a Spatio-temporal graph to record the RSS of each AP and only probes channels of neighbor APs in the following prescan operations. Variations of the RSS imply possible movement directions of the vehicle. During a handover, this scheme tends to select the AP whose RSS is increasing, so as to avoid wrong decisions.

In Adaptive Preemptive Fast Handover (APFH) [33], the STA may also send probe requests prior to a handover, and the APFH is adopted as the scanning strategy to reduce overheads and to increase throughput on the wireless channel. The STA will not start pre-scan unless the RSS of the currently serving AP drops below a defined threshold.

2.3.4 IEEE 802.11k and 802.11r for Handover Optimization

The 802.11k [19] allows STAs to request a neighbor report containing information about known neighbor APs that are candidates for handover. To facilitate handover, an 11k capable STA associated with an AP sends a request to a list of neighbor APs. The request is sent in the form of an 802.11

management frame, known as an action frame. The AP responds with a list of neighbor APs on the same WLAN with their WiFi channel numbers. The response is also an action frame. The STA identifies the APs candidates for the next handover from the response frame. The use of 802.11k radio resource management (RRM) process allows the STA to handover efficiently and quickly. To find an AP to handover from the neighbor list information, the 802.11k capable STA does not probe all of the 2.4 GHz and 5 GHz channels. Therefore, it reduces handover time and improves the decisions taken by the STA. However, the neighbor list is only recorded based on RSSI. As RSSI is considered not to be the best predictor for handover triggers [34], other parameters such as channel load, delay and STA QoS should be considered when the neighbor list is updated.

802.11r [20] uses Fast Basic Service Set Transition (FT) to allow encryption keys to be stored on all of the APs in a network. This way, a STA does not need to perform the complete authentication process. Thus, the authentication delay is reduced. However, it only supports the STAs association to WLANs which have 802.11r enabled.

2.3.5 Handover Decision Criterion

The handover decision time also affects the handover performance. Late handover execution may result in a long handover delay and service disruption. An early handover may force the handover execution to a new AP even when the link quality of service AP is still strong enough, resulting in a loss of the benefits of the preceding interface, which can include such factors as the bandwidth, QoS, and communication price [35].

The handover decision time can be affected by two aspects including the handover predictors and the threshold of the handover predictors. These two aspects determine when and how to trigger handover.

In the 802.11 standard [15], there are no specific predictors defined to trigger the handover. Most handover algorithms are based on RSSI to trigger a handover [30, 32, 33, 36]. Besides RSSI, the number of frame retransmissions or frame losses is also used as predictors for handover decisions [37]. A recent study by B. NG et al. [34] found that SNR, round-trip-time (RTT) and RX Bytes were better predictors than RSSI but they did not mention how to set the threshold in different network environments.

Usually, the threshold of the predictors is user pre-defined. However, pre-defined predictors are not suitable for dynamic network requirements. This is because several network parameters change over time such as the channel conditions, the STA speed, and transmit power and it is difficult to determine an optimal threshold in advance. Although some methods used link monitoring to predict a threshold [35, 38, 39], they only used one specific predictor such as RSSI or frame loss rate. In cellular networks, some researchers researched on the adaptive thresholds based on neighbor cells information [11, 39, 40]. However, the 802.11 networks lack neighborhood management. Therefore, it should be investigated how to dynamically adjust the threshold considering all the network predictors in order to improve 802.11 handover performance.

2.3.6 Summary of Reducing Handover Delay Solutions

Table 2.1 shows a summary of current solutions to reduce handover delay. From the survey of the literature above, few previous works have optimized the scanning parameters and the number of channels together to reduce both switching time and probe waiting time. Therefore, how to optimize all the affected scanning parameters together to improve handover performance is necessary for 802.11 networks. These parameters include *MinChannelTime*, *MaxChannelTime*, channel switching time, probe request interval, the number of channels scanned and channel sequence. The challenges are how to optimize these parameters according to the changes in the network environment.

Table 2.1: Summary of handover solutions

Work and Reference	Strategy	Contributions
A. Mishra et al. [3], [25], S. Shin et al. [4], C. Tseng et al. [5], IEEE 802.11k [19],	Neighbor Graph, Channel mask, Location-based, Neighbor report	Reduction of the Number of Channels
A. Mishra et al. [26], [27] [28], G. Castignani et al. [6], R. Pazzi et al. [7], A. Boukerche et al. [8], A. Arcia-Moret, et al. [29],	Fixed scanning parameters, Adaptive scanning parameters, Dynamically adjust parameters, Self-congured parameters, Sorting channels by Cultural Algorithm	Optimizing scanning parameters
I. Ramani et al. [30], YS.Chen et al. [32], V.M.Chintala et al. [33],	SyncScan, DeuceScan, Adaptive Preemptive Fast Handover (APFH)	Designed pre-scan schemes
IEEE 80211.r [20],	FastTransition(FT) authentication protocols,	Reduce anthentication delay
B. NG et al. [34],	Measurement and multiple regression	SNR, round-trip-time (RTT) and RX Bytes are better predictors than RSSI
V. Mhatre et al. [35], G. Athanasiou et al. [38], S.-J. Yoo et al. [39],	Link monitoring,	Predict thresholds

2.4 SON Functionalities

In the 3rd Generation Partnership Project (3GPP) current standards of mobile communications, such as LTE (Long Term Evolution) and LTE-Advanced, automated features is one of the key elements. Networks with those automated capabilities are known as Self-Organizing Networks (SON). SON is a promising automation technology used to simplify planning, configuration, optimization and healing. SON has been defined as a set of principles and concepts to add automation to mobile networks requiring less maintenance than traditional networks while improving Quality of Service (QoS) [41]. The SON functionalities are classified into three groups: self-configuration, self-optimization and self-healing [42].

In the following sections, the details of the three SON functionalities used for handover optimization in cellular networks and the challenges of SON applied in WiFi networks will be discussed.

2.4.1 Self-configuration

The aim of the self-configuration function used in WiFi handover is to automatically establish and update a neighbor list. This is because the most important phase of handover is to select the target AP from neighbor APs. In WiFi networks, there is no neighborhood management such as the Neighbour Cell List (NCL) in cellular networks. The STAs have to scan all the channels to find the target AP when a handover is initiated. This can lead to degradation of the STAs' throughput because of the additional delay for scanning the whole bandwidth.

Although the standard 802.11k can provide information about neighbor APs, this information is generated on-demand and is unable to be maintained. There are also potential neighbor APs that may not be provided by a neighbor report because the neighbor reports are based on a STAs moving path and only the neighbor APs on the moving path of STAs can be heard. The lack of neighbor AP information may lead to handover failure [43]. On the other hand, if the STAs scan an excessive number of neighbor APs, the time to find the best candidate for handover is significantly increased. It also results in a reduction in the throughput of STAs, and lowered QoS due to the more frequent occurrence of measurement gaps in data transmission [44].

Therefore, the neighbor list should be maintained and managed to provide better handover performance. However, it is a challenge to manually maintain and manage the neighbor APs in traditional WiFi networks.

Thus, a self-configuring neighbor list that can be automatically managed is necessary.

However, it is hard to automatically manage the neighbor relations in WiFi networks. In 2G/3G, there is a base station controller (BSC) or a radio network controller (RNC) used to connect with base stations so that the controller can get the information of each base station to configure the neighbors [45, 46]. In addition, LTE networks can easily communicate and broadcast neighbor cells' information [47]. In particular, the Automatic Neighbor Relation (ANR) technique can automatically optimize the neighbor list [48]. However, in WiFi networks, it is difficult for APs to communicate and cooperate. Although there is some research on generating neighbor relationships using a neighbor graph [3, 25, 49], the neighbor graph was based on the IEEE 802.11f standard, also known as the Inter Access Point Protocol (IAPP) [50], which has been withdrawn from the 802.11 standard since 2006. Also, the calculation time for producing a neighbor graph is too long to meet the dynamic network requirements.

Recently, IEEE amendments 802.11k [19] have been added to the IEEE 802.11 standard to enable STAs to obtain the neighbor AP's information by requesting a neighbor report. However, the neighbor information is generated on-demand and based on the STAs moving path. Only the neighbor APs on the moving path of STAs can be heard and the potential neighbor APs that cannot be provided by the neighbor report. The standard 802.11r considers configuration and data exchange for fast transitions, minimizing service interruptions. However, the standard 802.11r [20] still does not mention how to manage neighborhood relations. Therefore, only using these amendments is not suitable for managing the neighbor relations in WiFi networks [41]. It is still much harder to automatically manage the neighbor list in WiFi networks than in cellular networks because there is no inter-AP communication protocol and no centralized controller to manage APs.

Table 2.2 and 2.3 show a summary of some approaches about how to

Table 2.2: Summary of research about establishing and updating neighbor list in wireless networks

Work and Reference	Architectures	Techniques	Contributions	Drawbacks
3GPP TS32.500 [48]	Centralized	ANR	Automatically establishing neighbor relations for newly deployed cells	Is not suitable for Handover
F. Parodi et al. [51]	Centralized and distributed	Using geographic and antenna information	Automatically establish initial neighbor list for a new deployed Base Station	Is not suitable for Handover
A. Mishra et al. [25]	Distributed	802.11f-IAPP, capturing the mobility topology using the reassociation messages	Automatically establish and update neighbor relations by neighbor graph, reducing reassociation delay	802.11f-IAPP was withdrawn from the 802.11 standard, long computation time to capture the neighbor graph
A. Mishra et al. [49]	Centralized	Human observation of the reassociation messages	Automatically establish and update neighbor relations using the neighbor graph, reducing authentication delay	Lacking potential APs only by human observation
M. Shin et al. [3]	Distributed	802.11f-IAPP	Automatically establish and update neighbor relations using the neighbor graph, reducing scanning delay	802.11f-IAPP was withdrawn from the 802.11 standard, long computation time to capture the neighbor graph

Table 2.3: Summary of research about establishing and updating neighbor list in wireless networks - continued

Work and Reference	Architectures	Techniques	Contributions	Drawbacks
Y. Zhang et al. [52]	Distributed	Installing extra WiFi interfaces to broadcast information	The neighbor relations are manually established or automatically established based on reassociation messages, define the handover threshold by the value of signal quality	Making changes of APs, cannot automatically update the neighbor list efficiently, signal quality is the only predictor for handover, the threshold is not dynamic value
H. Zhang et al. [53]	Centralized	Periodically collects RSSI of neighbor AP using a full scan	Create and update neighbor list based on RSSI value, define the handover threshold using RSSI	The full scan takes a long time, RSSI is not the only predictor for handover, the threshold is not dynamic value
B. Zhang et al. [54]	Centralized	Periodically collects RSSI of neighbor AP using a full scan	Create and update neighbor list based on RSS value, defined scanning threshold and handover threshold	The full scan takes a long time, RSSI is not the only predictor for handover, the threshold is not dynamic value
S. Pack et al. [55]	Distributed	Capturing the mobility topology using reassociation messages	Create neighbor list using neighbor graph [25], select and update the neighbor list using neighbor weight, define neighbor weight threshold to select neighbor APs	Long computation time to capture the neighbor graph, neighbor weight threshold is not dynamic value
T. Lei et al. [56]	Distributed	Install wireless interfaces to get the wireless link information of neighboring APs	Create neighbor list by neighbor graph [25], sorted the neighbor list using predicted RSSI, both frame loss rate (FLR) and RSSI are handover threshold	Thresholds are empirical values

establish and update the neighbor list in wireless networks.

Firstly, two different network architectures are used to store the neighbor list. They are centralized architecture and distributed architecture. In the centralized architecture, the neighbor list is stored in the controller [48, 49, 51, 53, 54]. In the distributed architecture, the neighbor list may be stored in the AP [3, 25, 51, 52, 55] or the STA [56]. Both centralized and distributed architectures have their own advantages and disadvantages. In the centralized architecture, the neighbor list is maintained and managed consistently by a controller. The controller has enough storage space to save the neighbor lists. The controller can also give a global view to monitor the AP status and detect faulty APs. In the distributed architecture, extra wireless interfaces need to be installed to monitor and collect the neighbor AP's information. If the neighbor list is large, the storage space and computational complexity may not be acceptable to save and update all neighbor AP's information. The best candidate AP may not be found because of the lack of all neighbor APs information. Therefore, the centralized architecture is chosen in this thesis based on the assumption that there is no link delay between controller and AP.

Secondly, the handover threshold using different predictors is defined by different fixed values. Most of the existing research uses RSSI as the handover predictor. Other predictors including packet loss, SNR, delay, AP load also need to be considered in the WiFi handover decision. Moreover, the fixed values of the threshold are unable to meet the dynamic network requirement in different network topologies. Thus, the handover decision using dynamic threshold based on different predictors is necessary to select the best candidate AP from the neighbor list.

2.4.2 Self-optimization

The most important function of self-optimization is to enhance handover performance. As shown in Table 2.4 and 2.5, the handover performance

Table 2.4: Summary of research about optimizing handover parameters in wireless networks

Work and Reference	Method	Optimized parameters	Contributions	Drawbacks
D. Aziz et al. [57]	Genetic programming (GP)	Handover margin(HOM) and Time-to-Trigger (TTT) used for LTE handover	Minimize the number of handovers and dropped calls	Offline algorithm, long computing time, not suitable for real-time network conditions
V. Capdevielle et al. [58]	Simulated annealing (SA)	HOM, TTT and A3-offset used for LTE handover	Reduce unnecessary handovers and dropped calls	Limited updated parameters
A. Arcia-Moret et al. [29]	Cultural Algorithm (CA)	*MinChannelTime*, *MaxChannelTime*, channel sequence	Reduce scanning delay	channel sequence is sorted based on the information of last scan, cannot meet dynamic network requirements
P. Bhattacharya et al. [59]	Artificial Neural Network (ANN)	RSS, Traffic Intensity (TI)	Online algorithm, Fast handover based on high accuracy of handover decision time	Limited handover parameters are updated
M. Ekpenyong et al. [60]	ANN	Call block rate, call drop rate, carried traffic and signal strength	Online algorithm, predict handover decision time, reduce handover failure rate	Long training time cannot meet dynamic network requirements
M. S. Dang et al. [61]	Fuzzy logic	Received power level, user population and used bandwidth	Improved load balance of base station after handover	Designed for an indoor environment
S. S. Mwanje et al. [62]	Q-learning	HOM, TTT	Update handover parameters based on mobility changes	Does not consider load balance after handover

Table 2.5: Summary of research about optimizing handover parameters in wireless networks - continued

Work and Reference	Method	Optimized parameters	Contributions	Drawbacks
I. Balan et al. [63]	Weighted function	Signal strength, HOM, TTI	Reduce handover failure rate	Based on a previous analysis, unrealistic
G. Castignani et al. [6]	Adaption function	*MinChannelTime*, *MaxChannel-Time*	Increase AP discovery rate	Based on the information of last scan, cannot meet dynamic network requirements
R. W. Pazzi et al. [7]	Adaption function	*MinChannelTime*, *MaxChannel-Time*	Reduce scanning delay	Based on the information of last scan, cannot meet dynamic network requirements

may be improved by reducing the handover delay and handover failure rate, or optimizing the handover decision time by different thresholds.

As the handover threshold is dynamically updated by the neighbor list mechanism discussed in section 2.4.1, self-optimization aims to optimize the scanning parameters which accounts for the majority of handover delay [26] in WiFi. Currently, the value of scanning parameters are vendor pre-defined. Thus, the scanning delay is the same in every scan initiated by a STA even in different network conditions. The scanning time should be dynamically adjusted based on real-time network requirements.

Many Artificial Intelligence (AI) algorithms [57–61] have been used for handover optimization problems. Due to the high dynamic requirements of WiFi networks, online learning is suitable for optimizing the scanning parameters in response to the real-time network conditions. Most research in Table 2.4 and 2.5 only considers optimizing a few handover parameters to improve handover performance. However, the more parameters are optimized, the more network performance is improved. Some research [6,7,29] optimizes the handover parameters based on previous knowledge of network performance, which is unrealistic in real-time networks.

As WiFi networks have more dynamic and external influences than cellular networks, the optimization algorithm has to update the scanning parameters based on any change in the network environment. Compared with different algorithms reviewed in Table 2.4 and 2.5, an intelligent online algorithm that can optimize all the related scanning parameters is needed.

A Genetic algorithm (GA) has a robust search ability to find the optimal values of scanning parameters efficiently to meet the dynamic network requirements. Thus, GA is considered as a solution for the optimization problem and reviewed in the next section.

2.4.2.1 Related Work of Genetic Algorithm

In this section, a genetic algorithm (GA) is reviewed because GA is used to optimize the scanning parameters. The reason why GA is chosen to solve the self-optimizing scanning parameters problem is discussed in section 4.2. The following subsection will discuss the related work about different GA operators and how to choose for self-optimization in WiFi networks.

2.4.2.1.1 Selection In this thesis, the selection methods including Tournament Selection, Roulette Wheel Selection, Rank-based Roulette Wheel Selection are used to compare the handover performance.

Tournament selection is one of the most popular selection methods in the genetic algorithm due to its efficiency and simple implementation [64]. The efficient and simple algorithm is suitable to optimize the scanning parameters based on real-time network conditions. In tournament selection, n individuals are randomly selected from the larger population and the number of n is the tournament size. The selected individuals compete against each other, and the individual with the highest fitness generation population is selected.

In Roulette Wheel Selection, all the individuals in the population are placed on the roulette wheel according to their fitness value. Each individual is assigned a segment of the roulette wheel whose size is proportional to the individual [65]. The problem of this selection is that the individuals with the largest fitness probability have more opportunities to be selected. This selection cannot guarantee to find the best solution for scanning parameters in the entire population.

Table 2.6: The advantages, disadvantages and time complexity comparison of three different selection operators

Methods	**Advantages**	**Disadvantages**	**Time complexity**
Tournament Selection	Fitness scaling or sorting not required [66], efficient [64,67], suitable for small size problems [67,68]	Cannot guarantee to reproduce the best solution of scanning parameters for large size population [69].	O(n)
Roulette Wheel Selection	It gives a chance for all of them to be selected [70], stable [71]	As population converges, loss of selection pressure [72] and the optimal value of scanning parameters may not be found.	O(n^2)
Rank-based Selection	Prevents convergence happening too quickly, robust towards optimal solutions [73]	Populations must be ranked on every cycle, the ranking time increases the optimizing time so it is unable to meet real-time network conditions [66].	O($N \ln n$)

The rank-based election was introduced by Baker [74] to eliminate the disadvantages of Roulette wheel selection. In the rank-based selection schemes, the individuals in the population are ranked by their fitness and then selection probabilities are computed according to their ranks rather than fitness values. This selection provides high accuracy to find the best solution for optimized scanning parameters. However, the computation time is long because of sorting fitness values. Thus, it is not suitable for solving the self-optimization problem.

Several studies have been performed to compare the performance of different selection methods. The Table 2.6 shows the advantages, disadvantages and the time complexity of the three selection methods. However, none of these researchers tested the algorithm on WiFi handover performance.

For the self-optimizing scanning parameters problem in WiFi networks, tournament selection is more suitable to solve this issue than Roulette Wheel Selection and Rank-based Selection. This is because the tournament selection has a fast and efficient convergence rate to meet the real-time network requirements [64, 67]. Also, the number of scanning parameters is not big and tournament selection is suitable for this small size problem [67,68].

2.4.2.1.2 Crossover Crossover operators aim to explore solutions over a wider area. This wider area can provide opportunites to find better solutions for optimized scanning parameters. The solutions of the crossover depend on how effectively the crossover operation is performed.

How to choose a crossover operator depends on the representation types and GA performance demanded by different applications. There are three types of representation, which are Binary representation, Integer representation and Real representation. The following sections will discuss how to select the representation type for the scanning parameters.

Binary representation is where an individual is a string represented by zero and one. Any application that demands the representation of features as either present or absent or can be represented using a discrete parameter space can be naturally represented with the binary encoding scheme [75]. The scanning parameters are real values, thus the binary representation is not suitable to deal with continuous search spaces in the scanning parameters optimization problem.

When there are more discrete variable values that should be represented using binary encoding the length of a chromosome increases which in turn reduces the rate of convergence. This problem can be overcome by using integers for representing the chromosomes instead of a binary value [76]. However, the scanning parameters are not able to be coded using integer representation either. Therefore, the binary representation and integer representation are not suitable to optimize the scanning parameters.

Both binary and integer representations are unable to deal with continuous search spaces and the scanning parameters demand high precision [77] [78]. Real representation results in increased efficiency and precision [79]. Therefore, real representation schemes and associated crossover operators are considered to be used for the scanning parameters optimization problem.

As the crossover operator is chosen by the representation types, the real representation crossover including single-point crossover, N-points crossover and uniform crossover are used to optimize scanning parameters.

In WiFi networks, the scanning parameters need to be optimized based on real-time network conditions. The continuous search spaces are needed when the scanning is initiated periodically and high precision is required to make sure handover can be timely and successful. Thus, the crossover operator adopted real representation is suitable for the self-optimizing problem.

The handover performance using real representation crossover including single-point crossover, N-points crossover, uniform crossover are discussed and compared in Chapter 4.

2.4.2.1.3 Mutation The mutation in GA is used primarily to maintain genetic diversity. The mutation operator along with a suitable crossover operator can make the overall search efficient [80] to find the most optimal values of scanning parameters. There are several types of mutation such as Random mutation, Boundary mutation, Non-uniform Mutation and Normally distributed mutation.

The random mutation operator [81] replaces the value of the chosen gene with a random value (uniform probability distribution) selected between the user-specified upper and lower bounds for that gene [82]. Therefore, the Random mutation ensures that the GA can search the solution space freely to find a better solution for the scanning parameters [83].

In boundary mutation, the value of the selected gene is randomly replaced with either the upper or lower bound [84]. As the gene represents a scanning parameter, genes mutated by boundaries cannot meet the diversity of scanning parameters in GA. It may lose the opportunity to find the best optimal scanning parameters for each channel and affect the reducing of scanning delay.

A non-uniform mutation is one of the commonly used mutation operators in real coded GAs [81] . This mutation operator tunes the upper or lower bound as the GA generation increases. The solution also depends on the size of the population. The search of this operator was performed uniformly at the beginning and very locally towards the end in order to reach equilibrium between exploration and exploitation [85]. This operator works better in a larger population with more generation. However, the time consumption is not required in dynamic network environments. Therefore, it is not suitable to optimize the scanning parameters which needs an efficient algorithm to update the values of scanning parameters to meet the real-time network requirements.

2.4.3 Self-healing

The fault management includes detecting faults in the network, diagnosing and fixing the problem. The manual troubleshooting of a cellular network is a complex task as it is time-consuming and laborious. Therefore, the self-healing aims to automatically operate the overall fault management process including the collection of information about network performance, detection and diagnosis of faults and compensation or recovery actions.

One of the fundamental cases used in self-healing for cellular networks is Cell Outage Management (COM). COM comprises Cell Outage Detection (COD) and Cell Outage Compensation(COC).

COD aims to automatically detect outage cells when the cells are not operating normally due to possible failures, or external failures such as power supply or network connectivity, or even misconfiguration.

On the other hand, COC aims to reduce the degradation caused by a failure in a cell until the fault is solved. The compensation for network degradation can be made by modifying the configuration parameters. These parameters are usually from neighbor cells. All the modified parameters will be reverted when the network failure is solved [86].

Thus, the self-healing of COM processes should contain:

- Alarm correlation, a diagnosis process triggered alarms that try to find the root cause of the alarm. If a is discovered, an automatic recovery action can be started to try to resolve the problem.

- Sleeping cell or cell outage detection, locate cells that do not transmit faults/alarms but still do not perform as planned.

- Cell outage compensation; reconfigure neighbor cells temporarily to compensate for the failure of another cell.

Many research works use machine learning algorithms to solve COM problems. In the literature [87–93], the machine learning techniques (e.g., K Nearest Neighbor(KNN), local outlier factor(LOF)) are applied to COD. The KNN and LOF algorithms for cell outage detection are compared in [89]. It is observed that KNN outperforms LOF in speed and reliability since LOF can sometimes misclassify normal cells. Most of these works are based on the analysis of the performance of the problematic cell or its neighbor cells. With this methodology, only the most severe cases can be detected but many other outage situations may go unnoticed.

Several COC algorithms have been proposed in the literature [94–99]. In these works, the authors present different algorithms based on the modification of one or more control parameters and using different techniques (e.g., Fuzzy Logic, Reinforcement Learning). With these modifications, the algorithm tries to force users in the cell edge of the faulty cell to move to a neighbor cell. As a consequence, the service area of the faulty cell is reduced. The main objective of COC algorithms is to reduce the degradation produced by cell outage.

In the self-healing function, neighbor cells in the NCL will be reconfigured and provide the coverage for the area of outage cells. After the self-healing process is finished, self-optimization can be executed to find optimal settings for the new network environment. Therefore, the three functions of SON including self-configuration, self-optimization and self-healing can work together automatically to adapt to the network conditions.

The summary of the comparisons of COD and COC algorithms in cellular networks is given in Table 2.7 and 2.8, respectively. These comparisons aim to show the key network performance metrics used in each approach. This gives a better understanding of the advantages and disadvantages of different approaches for how to implement self-healing in WiFi networks.

Self-healing in WiFi handover aims to keep the mobility connectivity in faulty networks. However, it is not easy to automatically diagnose and recover faulty networks. In WiFi networks, there is no network management protocol since the AP are provided by different vendors. Without the unmanaged neighborhood and network configuration protocol, the self-healing becomes even harder due to dynamics and random access operations. self-healing in WiFi networks is not the same as LTE networks where the outage can be divided into cell outage and site (eNodeB) out-

Table 2.7: Summary of self-healing COD using different machine learning algorithms in cellular networks

Solution	Method	Performance metrics	Advantages	Drawbacks
Supervised learning	KNN [87]	RSRP, SINR	Simple, efficient to detect faulty element, handover can be achieved based on real-time measurement	Hard to predict the number of clusters ′′k″ in large data sizes, so the accuracy of detecting faulty elements is affected, handover failure rate increased because of low detection accuracy
Unsupervised learning	DAP [92]	RSRP, RSRQ, CQI	Automatically classifying clusters can meet dynamic network requirements	Quadratic computational cost means need long computation time, cause handover failure because of detection latency
Unsupervised learning	K-means [91]	RSRP, RSRQ, CQI, handover attempts	Simple, efficient to detect faulty element, handover can be achieved based on real-time measurement	Hard to predict the number of clusters ″k″ to detect the faulty element, causes handover failure
Unsupervised learning	LOF [88]	inHO	Simple, efficient to detect faulty element, handover can be achieved based on real-time measurement	Misclassifies normal cell, handover failure because of lower detection accuracy
Unsupervised learning	Neural Networks [93]	RSRP, RRC, DCR, HO, BCR	High accuracy, the faulty elements may be detected correctly	Complicated, the long training time means handover cannot perform timely or handover failure

Table 2.8: Summary of self-healing COC using different machine learning algorithms in cellular networks

Solution	**Method**	**Performance metrics**	**Advantages**	**Drawbacks**	**Objective**
Heuristic	Immune Algorithm [94]	SINR	Fast compensation made handover perform efficiently	Sensitive to initial parameters can lead to handover failure	Coverage and quality
Heuristic	Genetic Algorithm [100]	Capacity	Immunity to initialization means the handover can be achieved without the initialization effect	Handover may fail in large data size because of long computation time	Capacity unilization
Supervised learning	Neural Networks [99]	Bandwidth signal, servers status signals	High accuracy means handover may be completed successfully	Complicated which means handover may be failed if the parameter setting is wrong	Spectral efficiency
Reinforcement learning	Fuzzy-logic based RL [97]	SINR	Online learning means handover may be achieved based on real-time measurements	Complicated means handover may be failed if the parameter setting is wrong	Coverage

age. In case of cell outage in LTE networks, the data can also be collected from the serving eNodeB or neighbor cells. However, the outage in WiFi can only occur at site level. If the AP is faulty, it is impossible to collect measurement data by the serving AP and even neighbor APs without neighborhood management.

2.4.4 Summary of SON in WiFi Handover

This section gives a summary of the needs and challenges for SON in WiFi networks.

The scanning delay in WiFi handover accounts for more than 90% of the overall handover delay [26]. One reason for this is that there is no neighborhood management in WiFi. The STAs have to scan all the channels to search for the candidate APs. Therefore, a self-configuration approach to establish the neighborhood of APs by relationship in a neighbor list is necessary. With the neighbor list, the number of channels scanned will be reduced and the handover decision time may be adjusted based on the neighbor information. Thus, the total handover performance will be improved by reducing the handover delay.

Moreover, after the self-configuring neighbor list, the scanning parameters also need to be automatically optimized. The scanning parameters of *MinChannelTime*, *MaxChannelTime*, probe request interval and channel switching time are not defined in the IEEE 802.11 standard. They are defined in APs provided by different vendors. These fixed values cannot adapt to dynamic network requirements. Therefore, the self-optimizing algorithm is necessary to adapt to the network requirements.

Furthermore, fault management in WiFi networks is solved by human intervention. When the fault is diagnosed, it is recovered by human being manually reset or replaced. It costs a long handling time to maintain the network. Without the timely recovery or compensation for the faulty elements, it leads to intermittent or no connectivity between APs and STAs. Therefore, an intelligent self-healing approach is needed for network maintenance in WiFi networks.

However, cellular and WiFi systems have fundamental differences in the media access layer protocols. The SON functions of the cellular system cannot apply to WiFi system directly.

The application of SON functionality is not straightforward due to the followings:

- The WiFi networks lack coordination among APs. The coordination may be done through a centralized controller for a specific deployment.
- In WiFi networks, there is no neighborhood management to provide neighbor APs information.
- APs provided by different vendors may support different management protocols and configuration of network parameters.
- The initial radio configuration is not available from an operators management system. This is usually implemented using pre-defined vendor-specific settings that may include radio enabling (2.4 GHz vs. 5 GHz band), channel selection, modulation and coding scheme (MCS) index, Tx power [41].
- Network selection is driven by an associated STA's communication manager without any influence by the AP.

The above differences pose significant challenges that need to be accounted for implementing SON capabilities in WiFi networks.

2.5 Performance Evaluation

In wireless networks, there are three fundamental methods to evaluate network performance and explore new technologies: mathematical analysis, simulation and real measurement. These three methods are discussed in the following subsections.

2.5.1 Mathematical Analysis

Mathematical analysis provides a generic way to get performance results in various conditions through a mathematical formulation. These conditions are also perfectly controlled so that mathematical analysis provides a clear relationship between inputs and outputs.

Chang et al. [101] used a Fuzzy-Logic model to reduce the number of active scan process for every channel. They produce a FitAP factor as a handover decision parameter. While the handover occurs, the Fuzzy-Logic mechanism could choose a suitable channel according to the FitAP factor and only one active scan process is performed. The handover delay is reduced by approximately 70%. In [29], a multi-objective optimization algorithm using the Cultural Algorithm (CA) to optimize scanning timers including $MinChannelTime$ and $maxChannelTime$ [29]. The results show that the adaptive scanning strategies better manage the performance trade-off and allow different application profiles to match with specific scanning latency. Many machine learning models [57–60, 62] were used to obtain the optimal solutions when the output of the mathematical models could not be easily obtained in dynamic network environments.

Although these mathematical analyses optimized handover algorithm to reduce handover delay and increase successful handover rate, they are based on many assumptions. The results may not be correct with respect to what may happen in the real world. The accuracy of the analysis has to be considered through the validity of these assumptions. The mathematical models can be validated either by simulation or testbed or a combination of both.

2.5.2 Simulation

Simulation is used when the output of mathematical model cannot be derived because of the large size of a model or dynamic network environments. Simulation tools need to be investigated and selected according to different research issues. Simulation results also need to be validated to ensure the accuracy of simulation models.

2.5.2.1 Selecting simulation tools

There are many popular simulation tools available that allow researchers to control the network environment and parameters. Simulation tools need to be investigated and selected according to different research issues.

The comparison of popular simulation tools used in the wireless network is given in Table 2.9. Some simulation tools are open-source such as NS-2, NS-3. The open-source network simulator has a free license. This implies that everyone can freely study the source code, modify it and distribute it. The commercial simulator needs to buy a license at a high cost, and it is not allowed to debug it. Some Commercial simulators are free for academic research, including OPNET, OMNET++ and MATLAB. In this research, new handover algorithms need to be implemented by coding and debugging. Therefore, open-source or academic free simulators are more suitable for this research.

Mehta et. al [102] studied the usage of simulation tools in wireless networks. Their study shows that NS2 is the most popular tool for MAC (17%) and routing (20%) experiments across a variety of wireless network types (ad-hoc, mesh, sensor and cognitive radio) [103]. Other simulation tools used in the research community for WiFi simulation include Qualnet and Omnet++, which are also less popular than NS-2. MATLAB is only a generic simulator that is not designed for wireless networks [104].

However, NS-2 is very slow with a long simulation run and consumes high memory usage [105–107]. This slow and high consuming memory

Table 2.9: Summary of simulation tools used in wireless networks

Simulators	License	Wireless network support	Language	Energy model support	GUI	Limitations
NS-2	Open source	WLANs, Ad Hoc	C++/ OTcl	Yes	Poor	High computation time, high memory consumption, long time to learn
NS-3	Open source	WLANs, Ad Hoc, Cellular, WSN	C++/ Python	Yes	Good	Long time to learn
OPNET	Academic free	WLANs, Ad Hoc, Cellular	C/C++	No	Excellent	The number of nodes supported is limited, high memory consuming, insufficient tutorials
OMNET++	Academic free	WLANs, Ad Hoc, Cellular	C++	No	Good	Slow when simulation time is long, high memory consumption
QUALNET	Commercial	WLANs, Ad Hoc, cellular	C/C++	Yes	Excellent	The number of nodes supported is limited, difficult installation on Linux
NETSIM	Commercial	WLANs, Ad Hoc, Cellular, WSN	C	Yes	Excellent	Only support Windows platform, energy management only support in IoT module
MATLAB	Academic free	WLANs, LTE	C/C++/ Java	No	Excellent	Slow when simulation time is long, poor programming practices

simulator is not suitable when the number of nodes increases in a dense network environment or large network topologies scenarios.

NS-3 was intended to replace the popular NS-2 simulator and introduces many features over its predecessor in order to become the leading network simulator [108]). One of the fundamental goals in the NS-3 design was to improve the realism of the models for highly accurate and scalable network simulation technologies [109]. Thus, NS3 is very easy to build realistic scenarios. The simulation performance of NS-2 and NS-3 is also compared in [110]. The result shows that NS-3 outperforms NS-2 by having far less memory usage and less execution time. Compared to other simulators, NS3 consumes the lowest amount of memory as the number of nodes increases [105–107]. Moreover, the energy model supported by NS-3 is very useful for the self-healing function to detect faulty APs. Therefore, NS-3 is the most suitable simulation tool for this research.

2.5.2.2 Wireless Channel Model in NS-3

Wireless network simulators have their limitations. Due to the abstracted physical layer modelling, simulators are often concerned about not providing reliable results compared with real measurement [111]. The wireless channel in NS-3 tries to model the real functionality of the standard 802.11.

In NS-3, the wireless channel is modeled by the class **YansWifiChannel** which works together with **WifiPhy** class. **WifiChannel** includes the helper class **YansWifiChannelHelper** [108]. Each channel is configured by two models including PropagationLossModel and PropagationDelayModel [112].

There are fifteen PropagationLossModel in NS-3. These models calculate the reception power considering the transmission power and position of transmitting and receiving antennas [112, 113].

1. Cost231PropagationLossModel: This model applies to urban areas

and evaluates path loss in suburban or rural open areas. The frequency extends to the range of 1500 MHz to 2000 MHz [108, 112].

2. FixedRssLossModel: This model sets a constant received power level independent of the transmit power. If this loss model is chained to other loss models, it should be the first model to avoid excluding the losses calculated by the other included models [112, 113].

3. FriisPropagationLossModel: This model is valid only for propagation in free space within the so-called far field region. However, this model in such conditions shall not be considered realistic [108, 112].

4. ItuR1411LosPropagationLossModel: This model is designed for Line-of-Sight (LoS) short range outdoor communication in the frequency range 300 MHz to 100 GHz [108, 112]. The model expresses its loss based on the sum of: loss of free space, the diffraction loss of the roof to the street and the reduction due to the multiple diffraction screens of buildings [113].

5. ItuR1411NlosOverRooftopPropagationLossModel: This model is designed for Non-Line-of-Sight (LoS) short range outdoor communication over rooftops in the frequency range 300 MHz to 100 GHz. This model includes several scenario-dependent parameters, such as average street width and orientation. It is advised to set the values of these parameters manually (using the ns-3 attribute system) according to the desired scenario [112].

6. JakesPropagationLossModel: This model is a deterministic model. It is used in cellular mobile communications [113].

7. Kun2600MhzPropagationLossModel: This model is for urban areas at a frequency of 2600 MHz [108, 112, 113].

8. LogDistancePropagationLossModel: This model implements a log distance propagation model. The reception power is calculated with

the logarithmic distance model [108, 112, 113].

9. MatrixPropagationLossModel: The propagation loss of this model is fixed for each pair of nodes and does not depend on their actual positions. This model should be useful for synthetic tests [108, 112, 113].

10. NakagamiPropagationLossModel: This model implements the fast fading Nakagami model which explains the variations in signal strength due to multipath fading. The model does not account for the path loss due to the distance travelled by the signal. In a simulation, it is recommended to using it in combination with other models that take into account the path loss [108, 112, 113].

11. OkumuraHataPropagationLossModel: This model is used to model open area path loss for long distance (i.e., $>$1Km). The original Okumura Hata model is designed for frequencies of 150 MHz to 1500 MHz. Almost all models are designed for urban areas. Therefore, the model cannot cover all scenarios in all frequencies [108, 112, 113].

12. RandomPropagationLossModel: This model considers a loss of random propagation, and it changes each time the model is called. As a consequence, all the packets (even those between two fixed nodes) experience a random propagation loss [108, 112, 113].

13. RangePropagationLossModel: The propagation loss of this model depends only on the distance in meters (range) between the transmitter and the receiver. The single MaxRange attribute determines the path loss. The receivers in or inside the MaxRange meters receive the transmission at the transmission power level. Receivers beyond MaxRange receive at power -1000 dBm (effectively zero) [108, 112, 113].

14. ThreeLogDistancePropagationLossModel: This model implements a

log distance path loss propagation model with three distance fields. This model is the same as LogDistancePropagationLossModel except that it has three distance fields: near, middle and far with different exponents [108, 112, 113].

15. TwoRayGroundPropagationLossModel: This model implements a loss propagation model with a direct ray in line of sight and a second one reflected to earth. This model does not give good results for short distances due to the oscillations caused by the constructive and destructive combination of the two rays [112, 113].

In NS-3, PropagationDelayModels are implemented in two models [112]:

1. RandomPropagationDelayModel: The propagation delay of this model is completely random, and changes every time the model is called. All packages, even those that are sent between two fixed nodes, experience a random delay.

2. ConstantSpeedPropagationDelayModel: In this model, the signal travels with constant speed, equal to the speed of light. The delay is calculated according to the positions of the transmitter and the receiver. The Euclidean distance between the antennas Tx and Rx is used.

In NS-3, the default 802.11 channel models are LogDistancePropagationLossModel and ConstantSpeedPropagationDelayModel [112].

The LogDistancePropagationLossModel is a very popular model used in wireless networks and can be used to predict the propagation loss for a wide range of environments, including free space, urban, inside buildings and obstacles in buildings [114, 115]. Moreover, LogDistancePropagationLossModel is an empirical model that is easier to implement, require less computational effort, and are less sensitive to the environment [114, 116]. Compared with RandomPropagationDelayModel, the order of the transmitted packets of ConstantSpeedPropagationDelayModel is maintained and needs less computation time. Therefore, the default models are chosen in this research.

2.5.2.3 Validation of Simulation Results

The credibility of the simulation results may be questioned if the simulation parameters were not correctly configured. Therefore, simulation verification is important to test the reliability of the simulation results.

The NS-3 simulation model can be validated by comparing simulation results with conventional handover methods and other approaches in the literature using the same parameters.

The simulation model also can be validated using different values of parameters by multiple runs. Each run uses a different random seed which is used for traffic generation. The average of each performance metric is taken after the total runs. The performance metric includes average handover delay, average packet loss rate and average throughput. The averages shown are reported with a confidence interval of 95.00% under the assumption that the averages are normally distributed. This statistical methodology in reporting the results was designed consulting good practices on wireless simulation [103, 117].

2.5.3 Real Measurement

Testbeds are used to do the real measurement on real hardware. Since the experiment uses real equipment, the results obtained are practically accurate.

Velayos et al. [27] measured handover time for different 802.11b cards and concluded that different STAs showed different performance. The measurement results showed that the scanning delay reduced by 20% when the scanning timers $minChannelTime$ and $maxChannelTime$ were set to 1ms and 10.24ms respectively. In [3], geometrical mathematical models are used to analyse the neighbor APs range to support the handover process by selecting an appropriate AP. After analysing the theory of neighbour graph algorithms, the author measures probe delay based on the neighbor graph scheme and the traditional method in a campus building. The neighbour graph algorithm was verified by results of the experiment which shows better performance than previous scanning schemes.

Altough the real measurement provides realistic results, it is very difficult to control the random issues caused by hardware, temperature, environment and solar radiation. In contrast, simulation can control the random behaviour of the system's components and environment conditions.

As the actual equipment may be expensive, only small-scale applications with a smaller number of nodes are involved. For economical experiments, a simulation is better than testbed because simulation can be carried out without the real hardware. Moreover, simulation is much easier to simulate a dense network with a large number of nodes and large network topologies.

2.5.4 Summary of Performance Evaluation

The evaluation of SON functionalities of WiFi networks can be done through mathematical analysis, simulation and real measurement. Simulation and testbed can be used to verify the accuracy of mathematical models based on some theory assumptions. However, it is difficult and expensive to set up experiments using testbed with a large number of nodes and multiple network topologies on campus. Compared with testbed, network scenarios are easily constructed and modified by simulation. More importantly, simulations can model large scale network topologies that could be very expensive [111]. Therefore, mathematical analysis and simulation are chosen for this research to evaluate the network performance.

2.6 Summary

This chapter reviewed the handover issues in WiFi networks and recent solutions solving the handover issues. SON functionalities in cellular networks have been introduced. Motivated by SON in cellular networks, SON approaches will propose to improve handover performance in WiFi networks. However, there are many challenges to implement SON functionalities in WiFi networks. The challenges was discussed in this chapter. In the following chapters 3, 4 and 5, these challenges will be considered to propose a SON approach to improve handover performance in WiFi networks. The methods used for evaluating handover performance are discussed in this chapter.

Chapter 3

Self-configuring Neighbor List

As discussed in Chapter 2, the scanning delay is the dominant component which accounts for more than 90% of the overall handover delay [26]. In this chapter, a self-configuring neighbor list mechanism is proposed to reduce the scanning delay in WiFi networks. The neighbor list is a data structure that tracks the neighboring APs suitability for accommodating a handover.

The optimized neighbor list reduces the number of scanned channels, the number of probed access points and the waiting time of probe responses of non-adjacent APs. Therefore, this optimization reduces the total handover delay and improves the handover success rate and Quality of Service (QoS) experienced by users over the WiFi network.

The neighbor list reduces the number of channels scanned by a STA and also provides the neighborhood relationship for self-healing discussed in Chapter 5 to compensate for the network performance degraded by faulty elements.

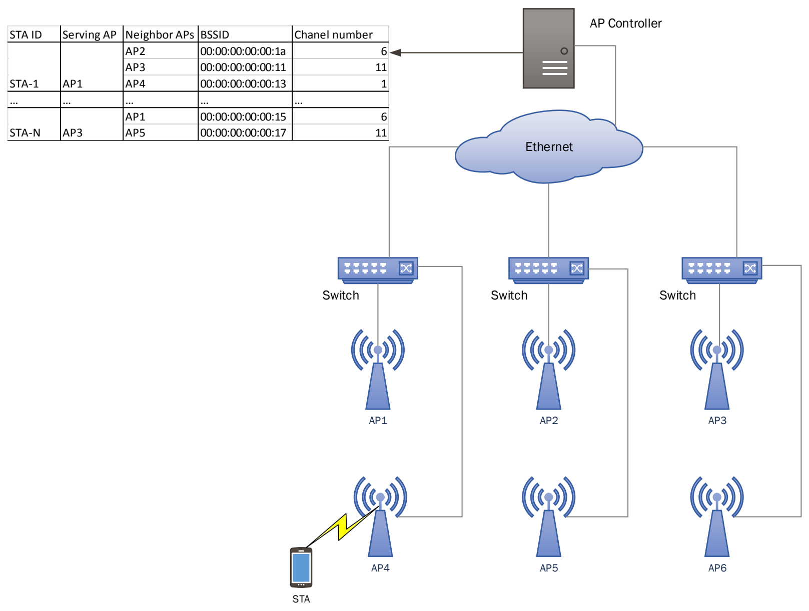
In the following sections, the proposed neighbor list mechanism is first introduced. Then, the details of the algorithms including Quick-sort and machine learning algorithms that are used to update the neighbor list are presented. Finally, the handover performance is evaluated by using these algorithms in different network scenarios.

3.1 Proposed Neighbor List Mechanism

As discussed in section 2.3, the STA has to scan all channels in order to find the best candidate AP in traditional 802.11 networks. This process involves too much unwanted probe waiting time of probe responses. This is because it is unnecessary to scan an AP that is not a neighbor of the serving AP. Without the neighbor AP's information, each STA associated with the same AP has to process the whole scanning phase to scan all channels when the handover is needed. In this case, if there is a neighbor list of each AP, the STA only needs to scan the channels of neighbor APs. Thus, the total scanning time can be reduced by the reduction of APs.

Although the neighbor report defined in IEEE 802.11k-2008 provides neighbor information, it does not define the mechanism for APs to retrieve the neighbor information. The neighbor information is generated on-demand and is not maintained in a controller. Also, the neighbor APs on the neighbor report are based on the STAs moving path. Thus, only the neighbor APs on the moving path of STAs can be heard so there are potential neighbor APs that cannot be provided by neighbor report. The neighbor information based on the neighbor report cannot be used in diverse network topologies.

In this proposed Neighbor List Mechanism (NLM), an up-to-date neighbor list is used to reduce the scanning delay. The neighbor list is stored in an AP Controller (APC) as shown in Figure 3.1. Each AP has its own neighbor list. The initial neighbor list can be obtained by an 802.11-based full scan and the 802.11k neighbor reports to get the neighbor AP information including BSSID, channel number and so on. The full scan and the 802.11k neighbor reports are used together to make sure all the available neighbor APs can be gathered by the first scan. After getting the initial neighbor list, the STA sends a probe request to the APC for the next scan. Once the STA starts a new scan, the APs on the neighbor list will be sorted based on the current radio and network environment. The highest pri-



STA ID	Serving AP	Neighbor APs	BSSID	Chanel number
		AP2	00:00:00:00:00:1a	6
		AP3	00:00:00:00:00:11	11
STA-1	AP1	AP4	00:00:00:00:00:13	1
...	...	...	...	...
		AP1	00:00:00:00:00:15	6
STA-N	AP3	AP5	00:00:00:00:00:17	11

Figure 3.1: NLM Architecture using an AP controller (APC) containing neighbor list information

ority AP on the neighbor list is considered as the best candidate AP for handover. With the reduction of probing APs and the number of channels scanned, the total scanning delay will be reduced.

The IEEE 802.11k-2008 standard provides mechanisms for gathering data including Received Signal Strength Indicator (RSSI), Signal to Noise Ratio (SNR), noise histogram request/report and BSS average delay. These data gathered on radio link performance and on the radio environment are called radio measurement reports. The capabilities of the radio measurement reports can be leveraged and retrieved as following:

1. Signal strength: RSSI or SNR. RSSI is the received signal strength reported from the STA's WiFi module. It is not the same as transmit power from the AP. Therefore, RSSI shows how well the STAs can hear a signal from the AP. Hearing the signal is important and useful for determining if the STAs has a link to the AP.

SNR is the Signal-to-Noise Ratio. A high SNR suggests that the intended communication signal is well differentiated from the interference in the vicinity.

2. Load: Channel load, traffic load. The channel load and traffic load can help resolve load balancing. This resolution will help optimize the resources used by distributing loads fairly across different APs and reduce the likelihood of overloading any single AP.

3. Delay: BSS access delay, transmit delay. These two parameters affect the handover delay and data disruption. Reducing these delays can maintain a good wireless connection and maintain ongoing data session during handover. According to IEEE 802.11k standard, when BSS Average Access Delay Elements are set to 1, the STA can measure and report BSS Average Access Delay information.

   The value of transmit delay can be obtained from the Transmit stream measurement report.

4. STA QoS: the throughput, error rates, packets loss rates and transmission delay are monitored to measure the overall QoS of the network.

5. Channel number: the neighbor report provides the channel number of each neigbhour AP.

After retrieving these values from the measurement reports, the APC generates an initial neighbor list. However, this initial list does not consider the packet loss, throughput performance, load balance and traffic demand. This is because the initial neighbor list only obtains the neighbor AP information based on RSSI. Thus, in NLM the neighbor list will be further updated periodically with SNR, load and delay taken into account when STAs start a new scan.

As illustrated in Figure 3.2 the STA does an initial scan to get each neighbor AP's information during the time T_1 and sends it to the APC.

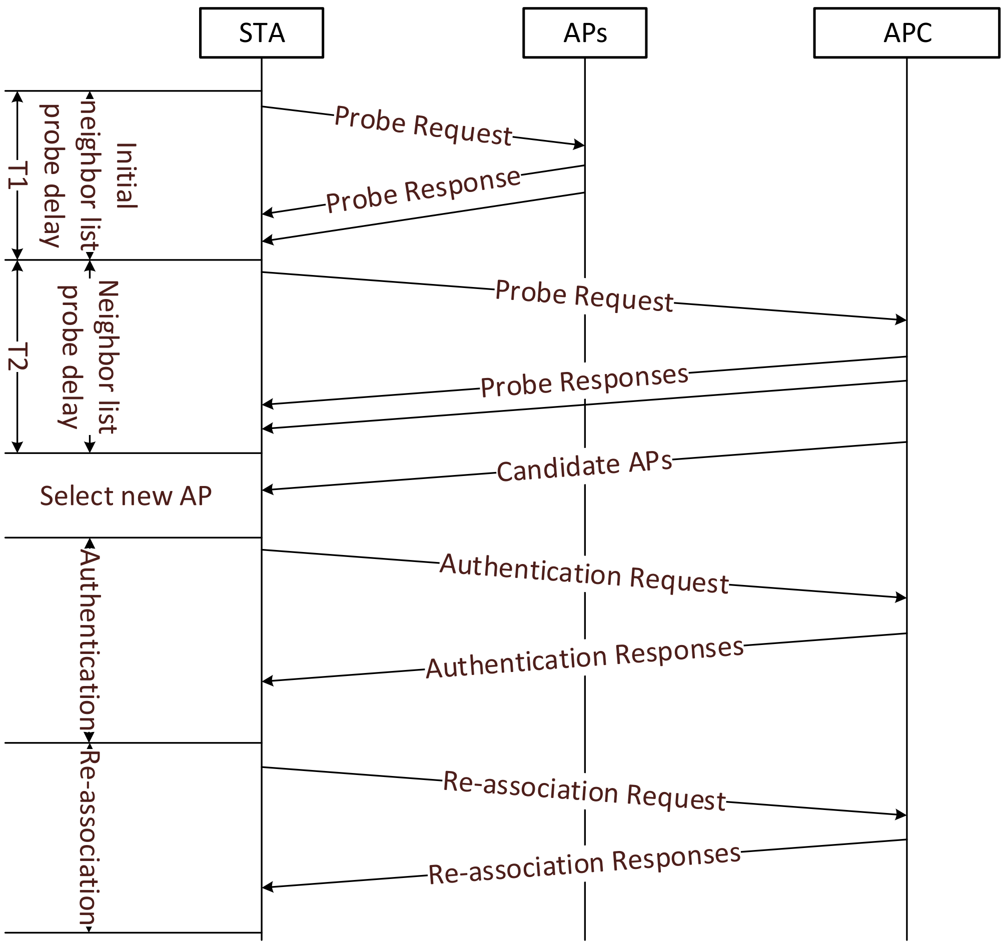


Figure 3.2: NLM Handover process using neighbor list and APC to reduce scanning delay

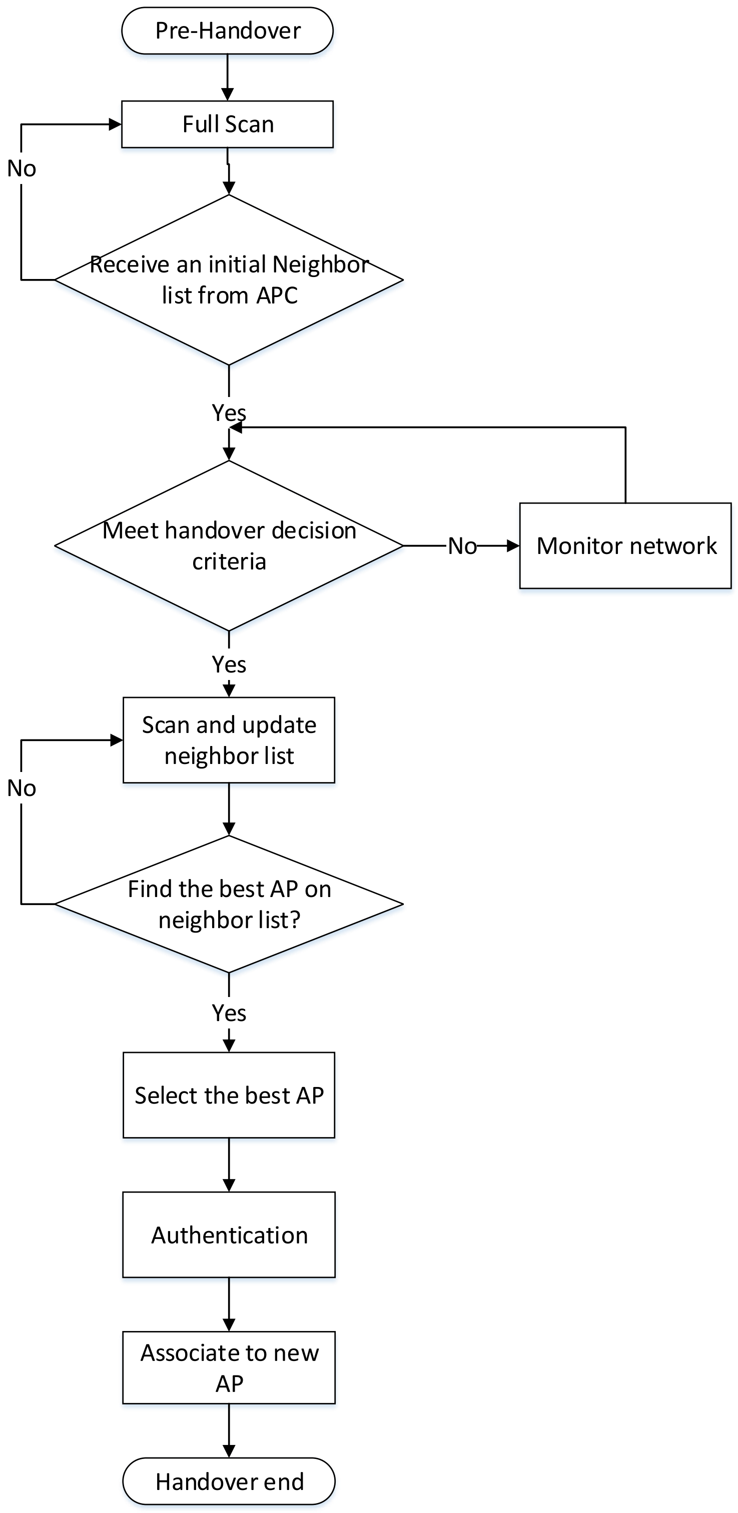

Pre-Handover
Full Scan
No
Receive an initial Neighbor list from APC
Yes
Meet handover decision criteria
No
Monitor network
Yes
Scan and update neighbor list
No
Find the best AP on neighbor list?
Yes
Select the best AP
Authentication
Associate to new AP
Handover end

Figure 3.3: Full flowchart for a STA to perform handover by using neighbor list and APC

For the following scan, each STA requests a neighbor list from the APC during the time T_2 and then the APs are sorted on the neighbor list based on the real-time radio measurement. After the STA obtains the updated neighbor list, the first AP on the neighbor list is chosen as the best candidate AP. The STA performs handover directly to the first AP without scanning more APs. After re-authentication and re-association with the new AP, the MAC layer handover process is completed. The completed NLM handover process is illustrated in Figure 3.3.

With the list, the scanning delay can be reduced and also the handover success rate will be increased. Moreover, the highest priority AP will be selected as the target AP for handover. The contribution of this approach is that it not only reduces the scanning delay, but also optimizes the handover performance. This is because the list generation is based on the mobility QoS and traffic demand, not only signal strength.

3.2 NLM Algorithms

The initial neighbor list is the input to the NLM updating algorithm and the output is a set of updated neighbors. The NLM problem can be formulated as follows: Let $P_i(n)$ denote the set of neighbors of the i-th AP whereby each neighbor is indexed by i such that $1 \leq i \leq n$ and $i \in \mathbb{Z}^+$. Using a set notation the neighbor set is expressed as:

$$P_i(n) = \{n_{i,1}, ..n_{i,k}, ...n_{i,n}\}, n_{i,k} \in \mathbb{N}, \tag{3.1}$$

where $n_{i,k}$ denotes the neighbor of the i-th AP with index k and $\mathbb{N}$ is natural numbers $\{0, 1, 2, ...\}$. The set of the neighbor index will be updated and sorted as a natural sorted set [118] (denoted by $P_i^o(n)$). The natural sorted neighbor list is sorted ascend based on the network metrics retrieved from measurement reports once a STA initiates a new scan. The updated algorithms of sorting neighbors can be quick-sort or machine learning algorithms, which are introduced in section 3.2.1 and 3.2.2. By updating the neighbor APs on the neighbor list, the neighbor APs are sorted according to real-time network conditions. The highest priority AP is considered as the best candidate AP for handover. The scanning delay is reduced because the STA does not need to scan all the channels to find the best AP. Thus, the total handover delay is reduced because the scanning delay is reduced.

3.2.1 Quick-sort

Algorithm 1 is the main function that defines the data structure to hold the i-th AP's information (lines 5 - 10, Algorithm 1), it also initializes three arrays (line 4, Algorithm 1) to hold RSSI, delay and packet loss measurements for each neighboring AP. The neighbor list updating work is invoked in line 19 of Algorithm 1 whereby a call to Algorithm 2 is made.

According to 802.11k-2008 standard, the neighbor APs can be obtained by the neighbor report. However, the number of neighbor APs is not defined in 802.11k-2008. As the deployment of AP depends on different network requirements such as AP coverage range and AP deployment cost [119], the size of neighbor list also depends on the AP deployment. In order to make sure the target AP for handover can be found on the neighbor list, the maximum size of the neighbor list is defined as 32 that is the same as specified by 3GPP in cellular networks [120] (lines 11, Algorithm 1).

In lines 6 to 8 of Algorithm 2, a natural sorting of the neighbors is achieved by a quick sort based on RSSI, packet delay and packet loss. After sorting, the position of each AP in the neighbor list is determined by the variable sum by line 9 in Algorithm 2, then AP priorities are sorted again based on position in line 10. Algorithms 3 and 4 show the partition and quicksort algorithms for generating natural sorting.

Algorithm 1 Update neighbor list main function

```
1: global variables
2:     global array $sum$;
3: end global variables
4: local array $a, b, c$;
5: struct AP
6: {
7: double $rssi$;
8: double $delay$;
9: double $loss$;
10: };
11: $NeighbourSize$ = 32 ;
12: struct AP $*arr$;
13: for $i = 1; i < NeighbourSize; i++$ do
14:     $sum[i]$ = 0;
15:     $a[i]$ = $arr[i].rssi$;
16:     $b[i]$ = $arr[i].delay$;
17:     $c[i]$ = $arr[i].loss$;
18: end for
19: UPDATELIST($a, b, c, NeighbourSize$);
```

Algorithm 2 Update AP priorities on neighbor list for a STA

```
1: function VOID UPDATELIST(array $a, b, c$, int $n$)
2:     $mid$ = $\lfloor n/2 \rfloor$;
3:     $pivot_{rssi} = a[0]_{rssi}$;
4:     $pivot_{delay} = b[0]_{delay}$;
5:     $pivot_{loss} = c[0]_{loss}$;
6:     QUICKSORT($a, 0, n-1, pivot_{rssi}$);
7:     QUICKSORT($b, 0, n-1, pivot_{delay}$);
8:     QUICKSORT($c, 0, n-1, pivot_{loss}$);
9:     POSITIONSUM($a, b, c, n$);
10:    QUICKSORT($sum, 0, n-1, a[mid]$);
11: end function
```

Algorithm 3 Partition the array of a, b , c and sum

```
function INT PARTITION(array $N$, int $left$, int $right$, int $pivot$)
    $m = (left + right)/2;$
    swap($N[m], N[left]$);
    $pivot = N[left];$
    $i = left + 1; j = right;$
    while ($i \leq j$)
        while ($N[j] > pivot$)
            $j = j - 1;$
        while ($i \leq j$ and $N[i] \leq pivot$)
            $i = i + 1;$
        if $i < j$
            swap($N[i], N[j]$);
            $i = i + 1;$
            $j = j - 1;$
    swap($N[i - 1], N[left]$);
    return $i - 1$
end function
```

Algorithm 4 Quicksort algorithm for neighbor list

```
function VOID QUICKSORT(array $N$, int $left$, int $right$, int $pivot$)
    if ($left < right$) then
        $q$ = PARTITION($N, left, right, pivot$);
        QUICKSORT($N, left, q - 1, pivot$);
        QUICKSORT( $N, q + 1, right, pivot$);
    end if
end function
```

Algorithm 5 Calculate AP position sum

```
function VOID POSITIONSUM(array $A, B, C$, int $n$)
    $i = 0$
    while ($i \leq n$)
        $sum[i] = 1/A[i] + 1/B[i] + 1/C[i];$
        $i = i + 1;$
end function
```

3.2.2 Machine Learning Based NLM

In the quick-sort algorithms, the APs on the neighbor list are sorted based on the position of each AP's parameters such as RSSI, delay or packet loss. With the quick-sort algorithms, the five algorithms (Algorithm 1 - 5) have to be executed whenever a STA requests a neighbor list. This cannot satisfy the dynamic network requirements. Therefore, an online algorithm that can sort the neighbor list immediately when a STA requests a neighbor list is necessary, because the faster the neighbor list is updated, the less scanning delay there is. By reducing the scanning delay, the data disruption time is reduced. The overall network performance will be improved by using online updating algorithms.

Learning to rank is a machine learning based ranking approach. It uses machine learning techniques to train a ranking model and uses the model online to get the ranking results for real-time problems. Therefore, learning to rank is a supervised learning task.

3.2.2.1 Supervised Learning Architecture

The proposed supervised learning is processed in APC. Once a STA initiates a handover process, the APC will send the first priority AP to the STA based on the supervised learning algorithms. In this thesis, the supervised learning algorithms used are linear regression, support vector regression(SVR)and artificial neural networks (ANN). The details of these algorithms will be introduced in sections 3.2.2.2, 3.2.2.3, 3.2.2.4.

As can be seen in Figure 3.4, the supervised learning process includes two phases that are the offline phase and the online phase. The offline phase aims to build up an initial learning model. This initial model is used for initialization, especially when the STA initiates a handover before the initial model is updated in the online phase. The initial model is trained based on the dataset collected by various network scenarios to ensure model accuracy. How to collect the offline data and normalize the

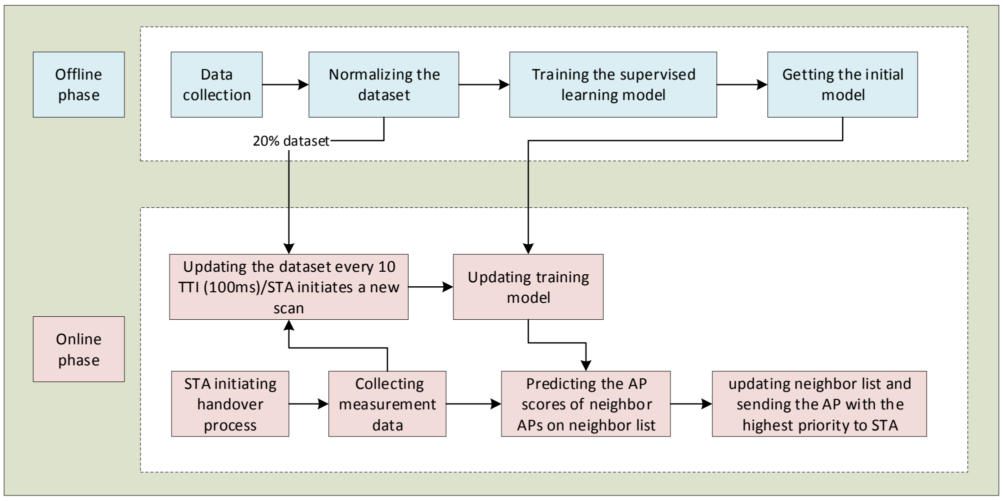

Offline phase
Data collection
Normalizing the dataset
Training the supervised learning model
Getting the initial model
20% dataset
Online phase
Updating the dataset every 10 TTI (100ms)/STA initiates a new scan
Updating training model
STA initiating handover process
Collecting measurement data
Predicting the AP scores of neighbor APs on neighbor list
updating neighbor list and sending the AP with the highest priority to STA

Figure 3.4: Supervised learning architecture including offline and online phases in APC

dataset is described in section 3.2.2.1.1 and 3.2.2.1.2 respectively. In order to obtain the initial model, the model has to be trained using the collected dataset and the parameters of the model are tuned to increase the accuracy of the model. How to validate and select the model is explained in section 3.2.2.1.3. After the model is selected, it is evaluated using the test dataset that is 20% of the collected dataset. The regression metrics presented in section 3.2.2.1.4 are used to evaluate the training model.

However, the initial learning model cannot meet the dynamic network requirements. Thus, online training is used to update the initial model based on real measurement data in the second phase. The real measurement data is used as input to replace part of the old dataset that is 20% of the offline phase dataset. The online dataset is updated every 10 TTI (100ms) or when a STA initiates a new scan. Then, the training model will be updated based on the new incoming measurement data. Finally, the neighbor list is sorted by AP scores, which are predicted by the new training model. The AP with the minimum AP score is chosen as the best candidate AP because the AP score label is ascendingly sorted in the training dataset(see section 3.2.2.1.1). The STA performs handover to the selected best candidate AP without scanning all the channels. The handover delay is reduced because the scanning delay is reduced. The handover performance using learning based NLM is evaluated by comparing with the 802.11 standard in section 3.3.4.

3.2.2.1.1 Offline Data Collection In order to obtain the initial learning model, a large set of training data is needed to train the model. The simulation setup is the same as section 3.3.3. The measurement data of four different network scenarios is collected. These four network scenarios include changing the speed of STAs, changing the number of APs, changing the number of STAs and changing the network topologies. Each scenario is run 50 times to collect the performance samples. Each sample is associated with one neighbor list of a STA. Table 3.1 shows the features of the dataset samples.

In the dataset, RSSI, SNR, packet delay, packet loss, data rate, average throughput and AP load are considered as the x vector of each neighbor AP. The AP score denoted by y is defined in Equat.14ion 3.7. The AP score is labeled based on the incoming handover (inHO) rate of each AP on the neighbor list because the inHO indicates the AP performance status. This is the statistical method used in cellular networks to monitor the cell status [121]. The inHO for each AP can be calculated using re-association response messages. The AP score is sorted ascendingly, which means the first AP on the neighbor list is the best.

After collecting and labeling all the data in the dataset. The dataset is shuffled randomly. Then, the dataset is divided into training and test sets with an 80-20 split. The training dataset is used for learning and training the model. The test dataset is used for testing the trained model.

3.2.2.1.2 Normalizing the Dataset As the network parameters are expressed with different units, the dataset used to predict and compare the

Table 3.1: Training database for machine learning

Measurements	Description	Unit
TTI	Transmission Time Interval	10ms
AP Identification (APID)	BSSID which is AP MAC address	Integer
RSSI	Received Signal Strength Indicator	dBm
SNR	Signal to Noise Ratio	dB
Packet delay	The time of data takes to reach the AP	s
Packet loss rate	Ratio of packets loss/total packet sent	percentage
Data rate	Speed of data transferred	bit/s
Average throughput	Average data packet from AP to STA per second	Mbps
AP load	Number of STAs connected	Integer
AP score	The score caculated by Equation 3.7, 3.27, 3.30	float

scores obtained on different scales will be normalized first. The benefit of normalizing data is that it eliminates the units of the measurement data. Another reason why normalizing is applied is that gradient descent converges much faster with normalizing than without it. The min-max normalization, is the simplest method and consists in rescaling the range of parameters to scale the range in [0, 1] or [-1, 1]. The general formula is given as:

$$x^{'} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{3.2}$$

where x is the value of the measurement data, and $x^{'}$ is the normalized value of the measurement data.

3.2.2.1.3 Model Selection and Validation

- **Model selection**

  Model selection is a process to choose different machine learning models such as linear regression, SVR or ANN, or different hyper-parameters or features for the same machine learning model such as deciding between different layers of neural networks, or different value of parameter γ in SVR.

  A successful model has the ability to be extended and generalized to unseen data. If a model has been trained too well on training data, it will be unable to generalize and it may make inaccurate predictions when given new data. This is called over-fitting. For example, in Figure 3.4, if the initial model is over-fitted, the prediction of AP score is not correct when new measurement data comes into the dataset. Using this wrong prediction to update neighbor list can cause handover failure. The inverse is also true. Under-fitting happens when a model has not been trained enough on the data. In the case of under-fitting, it also makes the model is not capable of making accurate predictions, even with the training data. Thus, the data collection in the offline phase in Figure 3.4 has to collect data that meet different network scenarios.

In this thesis, learning curves are used to estimate if a model is over-fitting or under-fitting. They show how the training and validation errors change with respect to the number of training examples used while training a machine learning model. By analysing these curves, a balanced model without over-fitting and under-fitting problems can be selected to predict the AP score of neighbor APs. The loss errors converge to small values as the training sample size increases in the balanced model such as the SVR model in Figure 3.7c. If the loss errors fail to decrease no matter how many training samples are in the training dataset, the model is suffering under-fitting such in Figure 3.7d. If a model has an over-fitting problem, increasing the training sample size decreases the training error but it fails to decrease the validation error, such as in Figure 3.7b.

- **Model validation**

  Cross-validation (CV) is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. It determines if a model is over-fitting or under-fitting. It is also used for evaluating different hyperparameters for machine learning algorithms. For example, in the case of training SVR, a range of γ is estimated and the value with the minimal validation error is selected. Therefore, CV is used effectively for model selection.

When the training set is split into k smaller sets, this approach is called k-fold CV. Each of the k ''folds" follows the procedures as below:

- A model is trained using k - 1of the folds as training data;
- the resulting model is validated on the remaining part of the data (i.e., it is used as a test set to compute a performance measure such as accuracy).

The final accuracy is measured by the average of the values computed in a loop. This method is suitable for a small dataset because the computation time is large to calculate k loops for a large dataset.

3.2.2.1.4 Regression Metrics The following regression metrics are used to evaluate the selected training model. The test dataset is 20% of the collecting dataset in the offline phase.

- **Explained variation regression score (EV)**: If $\hat{y}$ is the predicted target output, y is the corresponding (correct) target output, and has variance, the square of the standard deviation, then the explained variance is estimated as follow:

$$\text{EV}(y, \hat{y}) = 1 - \frac{Var\{y - \hat{y}\}}{Var\{y\}}. \tag{3.3}$$

  The best possible score is 1.0, the lower the EV values are, the worse the model is.

- **Mean absolute error (MAE)**: In statistics, MAE is a measure of the difference between two continuous variables. If $\hat{y}$ is the predicted value of the i-th sample, and y_i is the corresponding true value, then the MAE estimated over $n_{samples}$ is defined as:

$$\text{MAE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} |y_i - \hat{y}_i| . \tag{3.4}$$

- **Mean squared error (MSE)**: In statistics, the MSE or mean squared deviation (MSD) of an estimator (or a procedure for estimating an unobserved quantity) measures the average squared difference error between the estimated values and what is observed. The MSE is a measure of the quality of an estimator. It is always non-negative, and values that are closer to zero are better.

  If $\hat{y}$ is the predicted value of the i-th sample, and y_i is the corresponding true value, then the mean squared error (MSE) estimated over $n_{samples}$ is defined as

$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2. \tag{3.5}$$

- **R^2 score, the coefficient of determination**: In statistics, the coefficient of determination, denoted R^2, is the proportion of the variance in the dependent variable that is predictable from the independent variable(s). It provides a measure of how well observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model. The best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.0.

  If $\hat{y}$ is the predicted value of the i-th sample and y_i is the corresponding true value, then the score R^2 estimated over $n_{samples}$ is defined as

$$R^2(y,\hat{y}) = 1 - \frac{\sum_{i=0}^{n_{\text{samples}}-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n_{\text{samples}}-1}(y_i - \bar{y})^2}, \tag{3.6}$$

where $\bar{y} = \frac{1}{n_{\text{samples}}}\sum_{i=0}^{n_{\text{samples}}-1} y_i$.

3.2.2.2 Linear Regression Model for Updating Neighbor List

In this section, a linear regression model is introduced to update the APs on the neighbor list.

First, the dataset that is collected in the offline phase is split into training dataset and test dataset with an 80-20 split rule. The training dataset is used to train the linear regression model to fit the model parameters. The test dataset is used to evaluate the trained model using the regression metrics explained in section 3.2.2.1.4.

Let AP_i denote the i-th AP on the neighbor list. Let (x_i, y_i) denote a prepared dataset pair. y_i denotes the AP score that is a figure of merit for better handover performance. Let $(x_{i,1}, x_{i,2}, ..., x_{i,k})$ denote k network parameters measured by a STA using seven metrics such as RSSI, SNR, packet delay, packet loss, data rate, average throughput and AP load, and $\hat{y}_i$ denotes the predicted value by using the trained model and x_i vector value on the test dataset. As there is already an AP score label y_i from the test dataset, the predicted $\hat{y}_i$ will be compared with the original y_i to test the trained model accuracy by using the regression metrics expressed in Equation 3.3, 3.4, 3.5, 3.6.

As y_i is the dependent variable of the neighbor AP_i, the relation between the dependent variable (y_i in Equation 3.7) and the independent (or explanatory) variables ($x_{i,1}, x_{i,2}, ..., x_{i,k}$) are assumed to be linear. That is:

$$y_i = \omega_0 + \omega_1 * x_{i,1} + \omega_2 * x_{i,2}, +... + \omega_k * x_{i,k}, \tag{3.7}$$

where y_i the coefficients $\omega_0, \omega_1, ..\omega_n$ are the regression coefficients associated with $x_{i,1}, x_{i,2}, ..., x_{i,k}$ respectively.

Let $i = 1, 2, ..., n$, where n is the total number of APs on a neighbor list in the training data set. The n equations of the linear training model are obtained as follows:

$$\begin{aligned} y_1 &= \omega_0 + \omega_1 * x_{1,1} + \omega_2 * x_{1,2}, +... + \omega_k * x_{1,k}, \\ y_2 &= \omega_0 + \omega_1 * x_{2,1} + \omega_2 * x_{2,2}, +... + \omega_k * x_{2,k}, \\ &\vdots \\ y_n &= \omega_0 + \omega_1 * x_{n,1} + \omega_2 * x_{n,2}, +... + \omega_k * x_{n,k}, \end{aligned} \tag{3.8}$$

These n linear training model equations can be written as:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1k} \\ 1 & x_{21} & x_{22} & \cdots & x_{2k} \\ 1 & x_{31} & x_{32} & \cdots & x_{3k} \\ \vdots & \vdots & \ddots & \vdots & \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix} \begin{bmatrix} \omega_0 \\ \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \end{bmatrix} \tag{3.9}$$

The aim of this training model is to find the best coefficients ω_i which can successfully predict the value of AP score $\hat{y_i}$ with the smallest predict error. $\hat{y_i}$ can be expressed as follows:

$$\hat{y_i} = \omega_0 + \omega_1 * x_{i,1} + \omega_2 * x_{i,2}, +... + \omega_k * x_{i,k} + \epsilon_i, \tag{3.10}$$

where $x_{i,1}, x_{i,2}, ..., x_{i,k}$ are the test data set and $\hat{y_i}$ is the predicted score of each AP on a neighbor list. Let ϵ_i denote the random prediction error called the residual which is the difference between the value y_i on the test dataset and the predicted value $\hat{y_i}$ by the training model. The training model can be evaluated by using Equation 3.3, 3.4, 3.5, 3.6.

After training and testing the linear regression model, the trained model can be continuously learned and updated within a measurement period such as 10 Transmission Time Interval (TTI) that is 100ms in online training. When the new measurement data coming, the AP score will be automatically labeled according to the inHO rate of each AP. The training mode parameters will be tuned based on the dynamic network requirements.

Let $(x_i^{'}, y_i^{'})$ denote a real time measurement dataset pair. The predicted new AP score $y_i^{'}$can be expressed as follows:

$$y_i^{'} = \omega_0^{'} + \omega_1^{'} * x_{i,1}^{'} + \omega_2^{'} * x_{i,2}^{'}, +... + \omega_k^{'} * x_{i,k}^{'}, \tag{3.11}$$

where $\omega_0^{'}, \omega_1^{'}, ..\omega_n^{'}$ are the updated regression coefficients associated with new measured network parameters $x_{i,1}^{'}, x_{i,2}^{'}, ..., x_{i,k}^{'}$ respectively.

Using this online training model, a neighbor list with updated AP score can be obtained by a STA starting a new scan. Let $y_t^{'}$ denote the AP score of the target AP for handover. As the AP score is ascendingly sorted on the prepared dataset, the AP with the minimum AP score on the neighbor list is chosen as the best candidate AP. Thus, the target AP score $y_t^{'}$ can be expressed as follows:

$$y_t^{'} = \min\{y_1^{'}, ..., y_i^{'}, ..., y_n^{'}\} \tag{3.12}$$

where $\{y_1^{'}, ..., y_i^{'}, ..., y_n^{'}\}$ is the updated neighbor list for a STA, for example, in Figure 3.1, the neighbor list of STA-1 is $\{AP2, AP3, AP4\}$, $AP2$ has the highest priority AP and has the lowest AP score. Thus, $AP2$ is the best candidate AP for STA-1 to associate with.

3.2.2.3 Support Vector Regression for Updating Neighbor List

The linear regression has some drawbacks such as overfitting, which may occur when there are many features. A support vector regression (SVR) method is proposed to solve the overfitting problem. SVR is a new type of machine learning method which has good features such as higher fitting accuracy, fewer parameters, and global optimality. In real-time network environments, the relationship among network parameters may not be linear. SVR has been successfully applied to solve nonlinear systems and achieved good results [122]. Therefore, SVR model is chosen to compare with the linear regression model in section 3.2.2.2 to evaluate handover performance in 802.11 networks.

Let AP_i denotes the i-th AP on neighbor list. Let (x_i, y_i) denote a training data pair. y_i denotes the AP score that is a figure of merit for better handover performance. Let $(x_{i,1}, x_{i,2}, ..., x_{i,k})$ denote k network parameters measured by a STA using seven metrics such as RSSI, SNR, packet delay, packet loss, data rate, average throughput and AP load.

In the real network environment, the relationship between the network parameter x_i and AP score y_i is not linear. For nonlinear problems like this, SVR can be used to map the original data x into a higher-dimensional feature space by utilizing a nonlinear function $\phi(x)$ and then performs linear regression in the feature space

$$\hat{y} = f(x) = \omega^T \phi(x) + b, \tag{3.13}$$

where $\hat{y}$ is the predicted AP score by $f(x)$, $\phi(x)$ denotes a nonlinear mapping function from the input space x to a high-dimensional feature space, $\omega^T \in x$ is the transpose of weights vector, $b \in R$ is the bias.

The goal of ε-SVR [123] is to find a flat function $f(x)$ which can successfully predict the AP score $\hat{y}$. To ensure the prediction accuracy, SVR seeks to maximize the margin instead of minimizing the training error because the expectation is all the predicted values are within an ε-deviation of $f(x)$. As shown in Figure 3.5, if the predicted $\hat{y}$ between $f(x) - \varepsilon$ and $f(x) + \varepsilon$, the prediction is considered to have no error. Thus, to find the flat function Equation 3.13 means to seek the maximum margin, which means to seek the smallest value of the Euclidean norm $\|\omega\|^2$. Therefore, the SVR regression problem can be written as a convex optimization problem:

$$\text{minimize } \frac{1}{2}\|\omega\|^2 + \lambda \sum_{i=1}^{n} |y_i - f(x_i)|_\varepsilon, \tag{3.14}$$

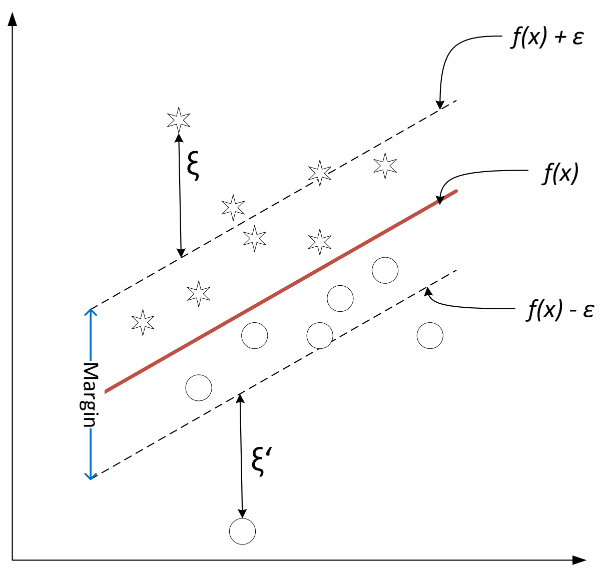


Figure 3.5: Margin between ε-insensitive zone and slack variables outside of ε-insensitive zone in support vector regression

where λ and ε are empirical parameters and $|y_i - f(x_i)|_\varepsilon$ represents the ε-insensitive loss function [124]. Let n denote the total number of APs on a neighbor list in the training dataset.

$$|y_i - f(x_i)|_\epsilon = \begin{cases} 0 & |y_i - f(x_i)| < \epsilon \\ |y_i - f(x_i)| - \varepsilon & \text{if } |y_i - f(x_i)| \geq \varepsilon) \end{cases}, \tag{3.15}$$

The value of the loss function is 0 when the predicted value of the error is less than ε; otherwise, a linear penalty is applied.

As the training dataset of the network parameters is unable to be completely separated without any errors (noise or outliers), positive slack variables are used to solve this issue and enhance the accuracy of the predicted value $\hat{y}$. The slack variables account for training data that fall outside of the ε-insensitive zone. A penalty parameter C is used to control how much error SVR is willing to afford in the training data. The best value of C is able to be found generally by CV.

By introducing the positive slack variables ξ_i and $\xi_i^{'}$, the minimization of Equation 3.14 is equivalent to minimizing the following constrained risk function:

$$\text{minimize } \frac{1}{2}\|\omega\|^2 + C(\sum_{i=1}^{n}(\xi_i + \xi_i^{'})) \tag{3.16}$$

subject to the constraints

$$\begin{aligned} y_i - \omega^T\phi(x_i) &\leq \varepsilon + \xi_i, \\ \omega^T\phi(x_i) - y_i &\leq \varepsilon + \xi_i^{'}, \\ \xi_i \geq 0, \xi_i^{'} &\geq 0, i = 1, ..., n, \end{aligned} \tag{3.17}$$

where $C > 0$ and is a real constant that represents a penalty for a prediction error that is greater than ε , and ξ_i and $\xi_i^{'}$ represent upper and lower constraints on the outputs of the model, respectively. For the soft-margin SVR using slack variables, the prediction error of the AP score, which need to be minimized, is proportional to C and the value of ξ_i and $\xi_i^{'}$.

To solve above optimization with constraints, the optimization can be converted to a quadratic programming problem by using Lagrangian multipliers. Thus, this constrained optimization problem can be expressed as the following Lagrangian function:

$$\begin{aligned} L := \frac{1}{2}\|\omega\|^2 + C(\sum_{i=1}^{n}(\xi_i + \xi_i^{'})) - \sum_{i=1}^{n}(\eta_i\xi_i + \eta_i^{'}\xi_i^{'}) \\ - \sum_{i=1}^{n}\alpha_i(\omega^T\phi(x_i) - y_i + \varepsilon + \xi) \\ - \sum_{i=1}^{n}\alpha_i^{'}(\omega^T\phi(x_i) - y_i + \varepsilon^{'} + \xi_i^{'}), \end{aligned} \tag{3.18}$$

where L is the Lagrangian function and $\eta_i, \eta_i^{'}, \alpha_i, \alpha_i^{'}$ are Lagrange multipliers, L follows from the saddle point condition that the partial derivatives of L with respect to $\omega, b, \xi_i, \xi_i^{'}$ have to vanish for optimality:

$$\begin{aligned} \frac{\partial L}{\partial \omega} &= \omega - \sum_{i=1}^{n} (\alpha - \alpha_i)\phi(x_i) = 0, \\ \frac{\partial L}{\partial \xi_i} &= \eta - (C - \alpha_i) = 0, \\ \frac{\partial L}{\partial \xi_i^{'}} &= \eta - (C - \alpha_i^{'}) = 0. \end{aligned} \tag{3.19}$$

As the SVR model needs to be trained fast to satisfy the real-time network requirements, the above problem is converted to a dual optimization which can be solved efficiently. At the same time, dual optimization can be easily cast as a convex quadratic optimization problem. The most important thing to use dual optimization is to introduce a kernel function, which can solve the non-linear neighbor list updating problem. By substituting Equation 3.18 into 3.19, the convex function in Equation 3.20 can be obtained.

$$Q(\alpha_i, \alpha_i^{'}) = \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^{'}) - \varepsilon(\alpha_i + \alpha_i^{'}) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i - \alpha_i^{'})(\alpha_j - \alpha_j^{'})K(x_i, x_j). \tag{3.20}$$

The solution can be obtained by maximizing Equation 3.20 subject to a new set of the constraints:

$$\begin{aligned} \sum_{i=1}^{n}(\alpha_i - \alpha_i^{'}) = 0, \\ 0 \leq \alpha_i \leq C, 0 \leq \alpha_i^{'} \leq C, i = 1, ..., n. \end{aligned} \tag{3.21}$$

With the Lagrange multipliers α_i and $\alpha_i^{'}$, the estimated output of AP score $\hat{y}$ can be represented by

$$\hat{y} = f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^{'})K(x, x_i) + b, \tag{3.22}$$

where $K(x, x_i)$ is called the kernel function. The kernel function can map the original training data into a space with a higher dimension to solve the non-linear neighbor list updating problem. According to the Karush-Kuhn-Tuckers (KKT) conditions of solving quadratic programming problem, only some of the coefficients, $\alpha_i - \alpha_i^{'}$, are not zeros, and the corresponding data vectors are called support vectors.

In SVR, a well selected kernel function determines how well the input data are mapped. Commonly used kernels include

- Linear kernel

  $$K(x_i, x_j) = x_i^T \cdot x_j, \tag{3.23}$$

  This linear kernel model can be used to solve a linear problem when the training data is small even with a large number of feature vectors. In the offline training, the training data is large to consider many network scenarios to ensure the accuracy of the model.

  However, the relationship between AP score and network parameters is not linear. Thus, the linear kernel is not suitable in neighbor list mechanism.

- Polynomial kernel

  $$K(x_i, x_j) = (\gamma(x_i^T \cdot x_j) + c)^d, \tag{3.24}$$

  where d is the degree of the polynomial and γ and c are constants.

  The polynomial kernel is usually used when feature vectors are not negative. It is a non-stationary kernel that is suited for problems where all the training data is normalized. In the neighbor list training phase, the data has been normalized through the RSSI value is negative. Thus, this polynomial kernel will be investigated in section 3.2.3.1.

- Gaussian RBF kernel with the following form:

  $$K(x_i, x_j) = \exp\left(- \frac{\|x_i - x_j\|^2}{\sigma} \right), \tag{3.25}$$

  where σ is a user-specified parameter and $\|\cdot\|$ denotes the distance between two input vectors.

  Gaussian RBF kernel is a general purpose kernel. It can be used when the training data and the number of feature vectors are both large. Therefore, the gaussian RBF kernel will be used and evaluated in the offline training phase for updating the neighbor list because the dataset and the number of network parameters are large.

- Sigmoid kernel

  $$K(x_i, x_j) = \tanh(\gamma(x_i^T \cdot x_j) + c), \tag{3.26}$$

  where γ and c are constant.

  The sigmoid kernel is often used as an activation function for artificial neurons in neural networks. The SVR using a Sigmoid kernel function is equivalent to a two-layer, perceptron neural network. This Sigmoid kernel will be investigated in neural networks to solve the neighbor list updating problem. The details of neural networks will be introduced in section 3.2.2.4.

Overall, a good way to choose the kernel function is to use k-fold CV, described in section 3.2.2.1.3, on the training dataset. Thus, these kernel functions will be evaluated in the model selection section 3.2.3.1 and a suitable kernel function for the SVR algorithm used for updating the neighbor list will be selected.

Let $(x_i^{'}, y_i^{'})$ denote a real time measurement data pair. Let $m = 1, 2, ..., n$, where n is the total number of APs on a neighbor list in the measurement dataset. When the SVR kernel model is selected, the predicted new AP score $y_i^{'}$ can be expressed as follows:

$$y_i^{'} = f(x_i^{'}) = \sum_{m=1}^{n} (\alpha_m - \alpha_m^{'}) K(x_i^{'}, x_m^{'}) + b^{'}, \tag{3.27}$$

where $\alpha_m^{'}, \alpha^{'}$ are the updated Lagrange multipliers and $b^{'}$ is the updated bias.

Using this SVR online training model, a neighbor list with updated AP score can be obtained when a STA starts a new scan. Let $y_t^{'}$ denote the AP score of the target AP for handover. As the AP score is ascendingly sorted on the prepared dataset, the AP with the minimum AP score on the neighbor list is chosen as the best candidate AP. Thus, the target AP score $y_t^{'}$ can be expressed as follows:

$$y_t^{'} = \min\{y_1^{'}, ..., y_i^{'}, ..., y_n^{'}\} \tag{3.28}$$

where $\{y_1^{'}, ..., y_i^{'}, ..., y_n^{'}\}$ is the updated neighbor list for a STA, for example, in Figure 3.1, the neighbor list of STA-1 is $\{AP2, AP3, AP4\}$, $AP2$ has the highest priority AP and has the lowest AP score. Thus, $AP2$ is the best candidate AP for STA-1 to associate with.

3.2.2.4 Artifical Neural Networks for Updating Neighbor List

In section 3.2.2.2, multiple linear regression is applied to the neighbor list updating. The correlation among network parameters including RSSI, SNR, packet delay, packet loss, data rate, average throughput and AP load are assumed to be linear. However, the correlation of these parameters is non-linear in the real world. A non-linear regression model using SVR is built in section 3.2.2.3 to solve the overfitting issue caused by linear regression. Also, when the kernel function is the Sigmoid kernel, the SVR model is equal to a two-layer Artificial Neural Networks (ANN) that has no hidden layers. In ANN, the number of nodes in a hidden layer, the number of hidden layers and the activation function used in a hidden layer will affect the prediction accuracy when the neighbor list needs to be updated because the learning time is longer when the number of nodes and hidden layers are larger. In real networks, the training time needs to fast enough to make sure the STA can find the best candidate AP. Usually, the ANN needs a large dataset to improve training accuracy, while SVR just selects the small size of the support vectors to train the model. The training time of SVR is faster than ANN. However, the offline phase needs to ensure the training model with high accuracy and a large dataset. Thus, the ANN needs to be compared with SVR for the whole learning phase to make sure that the neighbor list is updated accurately and timely.

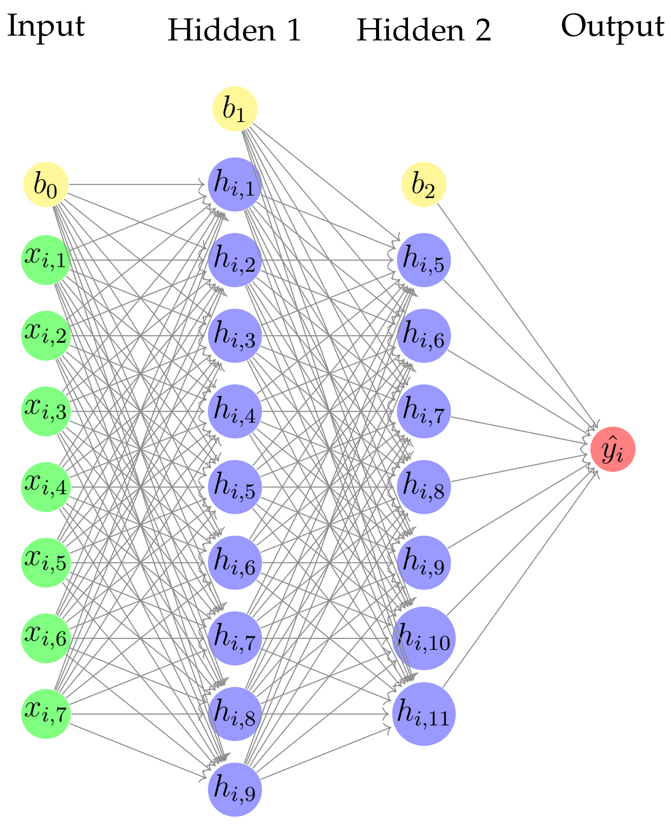


Figure 3.6: An example of four layers ANN model to predict the AP score of AP_i

In this thesis, a multilayer feedforward ANN algorithm is used to update the neighbor list. The multilayer feedforward ANN is chosen because it can solve the non-learning neighbor list updating problem more efficiently than the backpropagation neural network. This efficient algorithm is suitable for a real-time WiFi system.

In a multilayer feedforward ANN, the information moves in only one direction from the input nodes, through the hidden nodes (if any) and to the output nodes. The feedforward ANN contains one or more hidden layers (apart from one input and one output layer).

Figure 3.6 depicts the proposed ANN model for 802.11. Let (x_i, y_i) denote the training data pair. $x_i = \{x_{i,1}, x_{i,2}, ..., x_{i,k}\}$ denote the measurement information of AP_i such as seven network parameters including RSSI, SNR, packet delay, packet loss, data rate, average throughput and AP load. The output $\hat{y}_i$ is the predicted AP score of the i-th neighbor on the neighbor list. Thus, the ANN training model can be written as:

$$\hat{y}_i = \sum_{k=1}^{h_2} \omega_{3k} \sigma \left(\sum_{j=1}^{h_1} \left(\omega_{2j} \sigma \left(\sum_{i=1}^{n} \omega_{1i} x_i + b_0 \right) + b_1 \right) \right) + b_2, \tag{3.29}$$

where ω_{1i} is the weight vector associated with the input layer of n input nodes to the first hidden layer and b_0 is the bias at input layer for each node, ω_{2j} is the weight vector associated with the first hidden layer of $h1$ hidden nodes to the second hidden layer and b_1 is the bias at the first hidden layer for each node, ω_{3k} is the weight vector associated with the second hidden layer of $h2$ hidden nodes to the output layer and b_2 is the bias at the second hidden layer for each node. The activiation function is denoted as $\sigma(\cdot)$.

Let $(x_i^{'}, y_i^{'})$ denote a real time measurement data pair. The neighbor list with a updated set of AP score $y_i^{'}$ can be obtained by a STA starts a new scan. The predicted new AP score $y_i^{'}$ can be expressed as follows:

$$y_i^{'} = \sum_{k=1}^{h_2} \omega_{3k}^{'} \sigma \left(\sum_{j=1}^{h_1} \left(\omega_{2j}^{'} \sigma \left(\sum_{i=1}^{n} \omega_{1i'} x_i^{'} + b_0^{'} \right) + b_1^{'} \right) \right) + b_2^{'}, \tag{3.30}$$

Let $y_t^{'}$ denote the AP score of the target AP for handover. As the AP score is ascendingly sorted on the prepared dataset, the AP with the minimum AP score on the neighbor list is chosen as the best candidate AP. Thus, the target AP score $y_t^{'}$ can be expressed as follows:

$$y_t^{'} = \min\{y_1^{'}, ..., y_i^{'}, ..., y_n^{'}\} \tag{3.31}$$

where $\{y_1^{'}, ..., y_i^{'}, ..., y_n^{'}\}$ is the updated neighbor list for a STA, for example, in Figure 3.1, the neighbor list of STA-1 is $\{AP2, AP3, AP4\}$, $AP2$ has the highest priority AP and has the lowest AP score. Thus, $AP2$ is the best candidate AP for STA-1 to associate with.

3.2.3 Machine Learning Performance Evaluation

In this section, the training model will be selected based on the training dataset and loss errors. The selected model will be validated by the CV method described in section 3.2.2.1.3 and evaluated using the regression metrics explained in section 3.2.2.1.4.

3.2.3.1 Model Selection

The Figures 3.7a, 3.7b, 3.7c, 3.7d show the learning curves of SVR model with different kernel functions. In Figure 3.7b, there is a big gap between the errors. This model is suffering high variance, which is an over-fitting issue. In Figure 3.7d, the errors increase when the training examples are bigger. This model suffers under-fitting with high bias. Both the models in Figures 3.7a and 3.7c show the good performance with lower error.

When the SVR kernel="rbf", the training error is smallest. Therefore, kernel="rbf" is chosen for the SVR model.

Using the Gaussian RBF function, the value of γ is investigated as shown in Figure 3.8. When $\gamma = 0.00066$, the SVR model shows the best performance with the lowest loss error.

In Figure 3.9, three activation functions have been compared in ANN. When the activation = "tanh", the performance of ANN with the smallest error. Therefore, the activation function "tanh" is chosen in this thesis.

Table 3.2 shows the summary of the model selection based on the above analysis for these three machine learning algorithms.

Table 3.2: Regression metrics setting

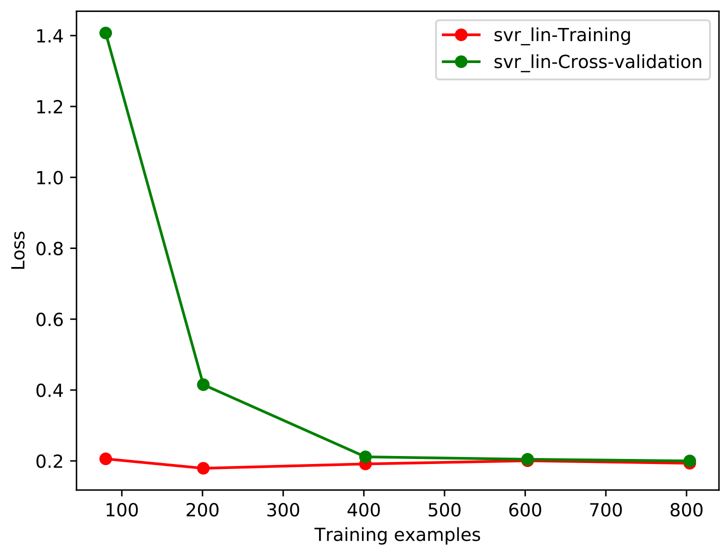
Algorithm	**Parameters**
MLR	Ordinary Least Squares
SVR	kernel="rbf",$\gamma = 0.00066, \epsilon = 0.1$
ANN	hidder layer=2, activation="tanh"

3.2.3.2 Model Validation

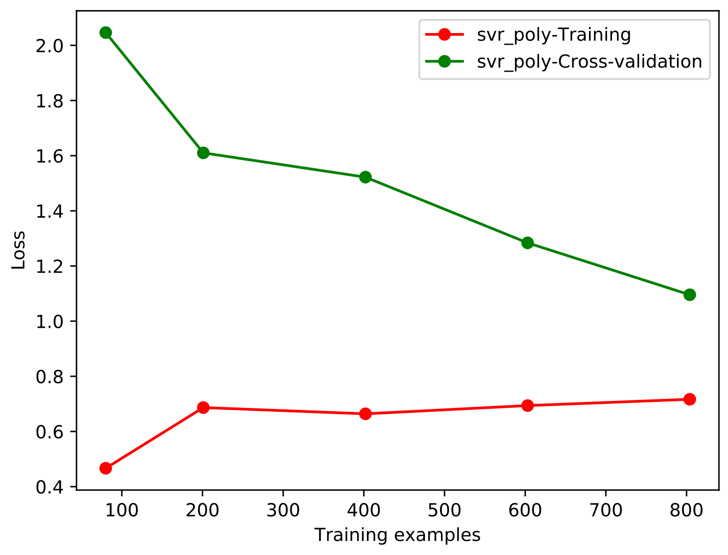
Table 3.3 shows the results using k-folds CV to validate the three models of MLR, SVR and ANN. The training data is split into five smaller sets. The mean value of SVR is the highest of these models after 5-folds validation, which means SVR is the best model to solve the neighbor list optimization problem.

Table 3.3: Cross validation result of the models inlcuding MLR, SVR and ANN

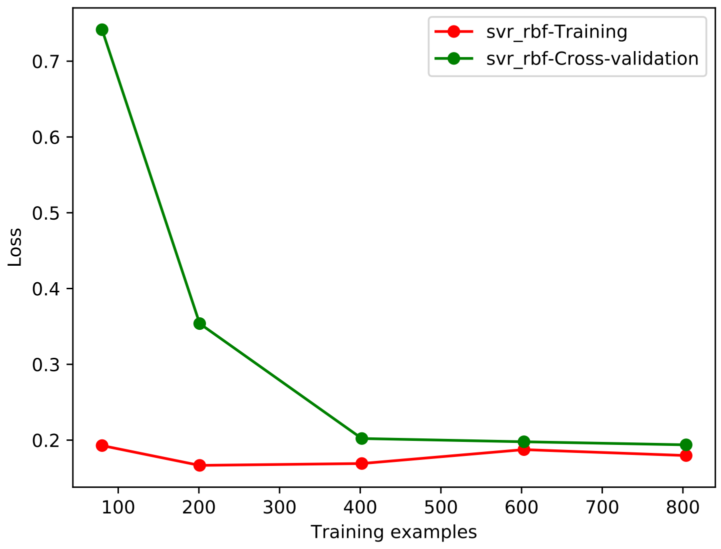
Algorithm	**k-folds**						
	0	1	2	3	4	5	Mean
MLR	0.927596	0.929659	0.868625	0.936785	0.930781	0.923459	0.919484
SVR	0.930621	0.933104	0.868497	0.930042	0.929930	0.931338	0.920589
ANN	0.930713	0.914467	0.842921	0.928515	0.895739	0.929035	0.906899



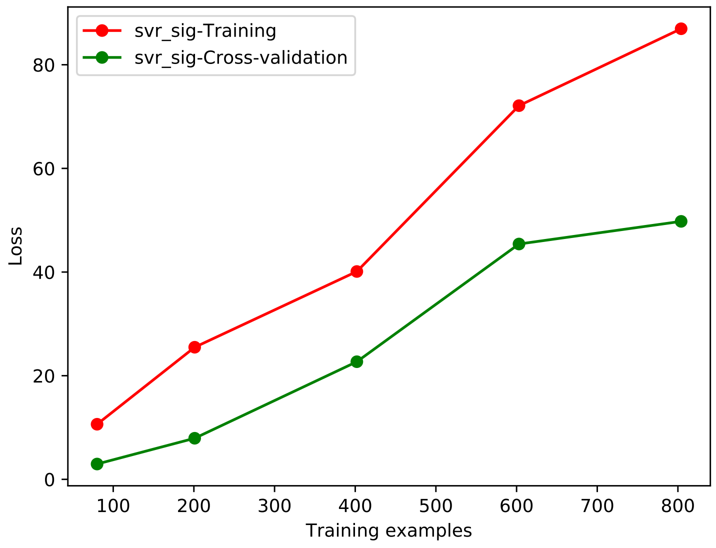
(a) SVR-kernel="linear"



(b) SVR-kernel="poly"



(c) SVR-kernel=”rbf”



(d) SVR-kernel=”sigmoid”

Figure 3.7: The learning curve for different SVR kernel and parameters to select SVR model

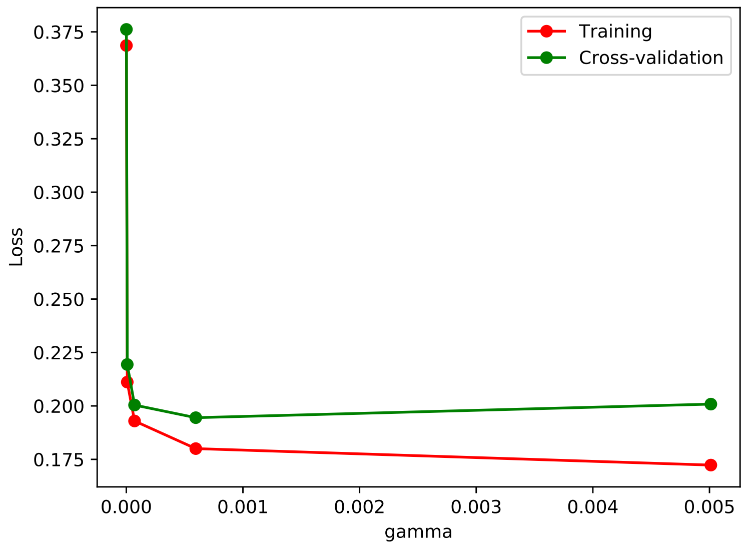


Figure 3.8: The value of γ for the SVR-kernel="rbf"

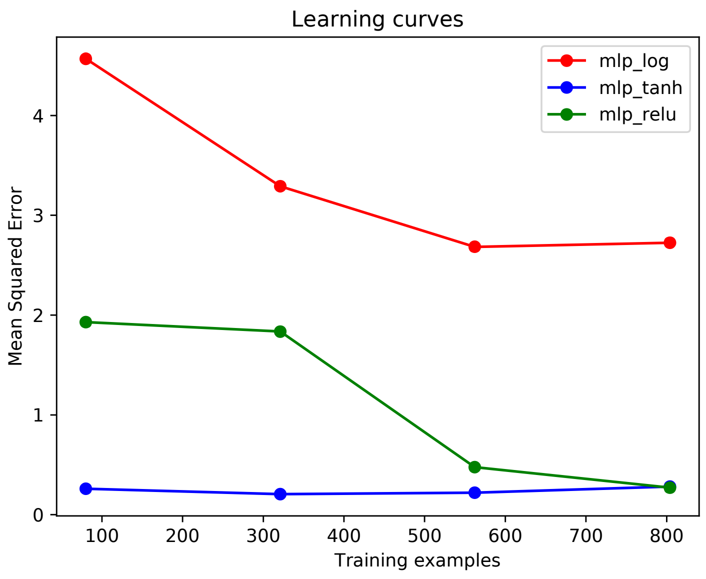
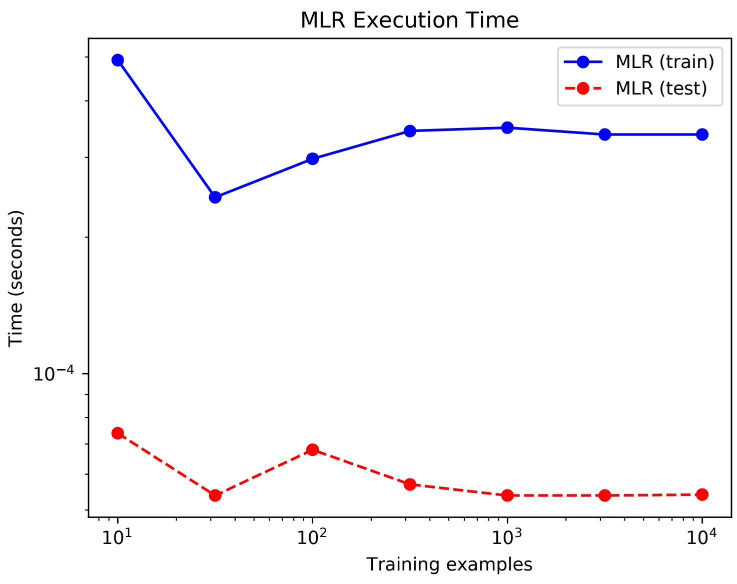
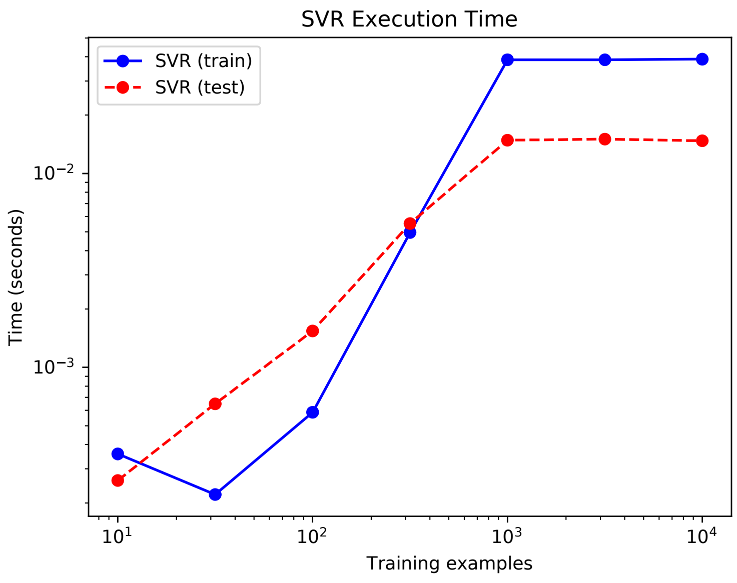


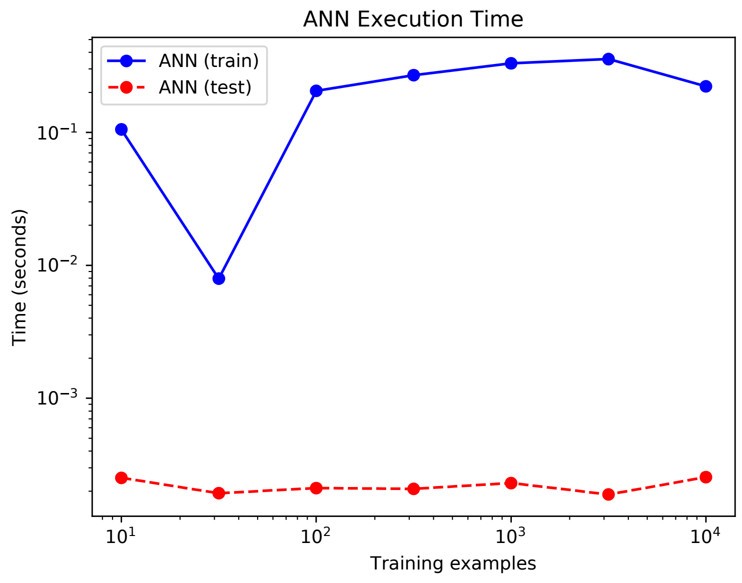
Figure 3.9: The learning curve of ANN activiation function



(a) Learning time of multiple linear regression

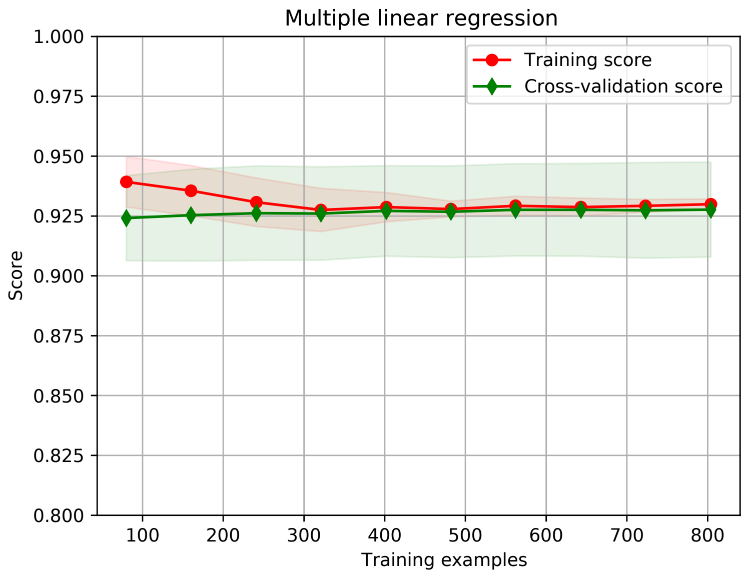


(b) Learning time of support vector regression

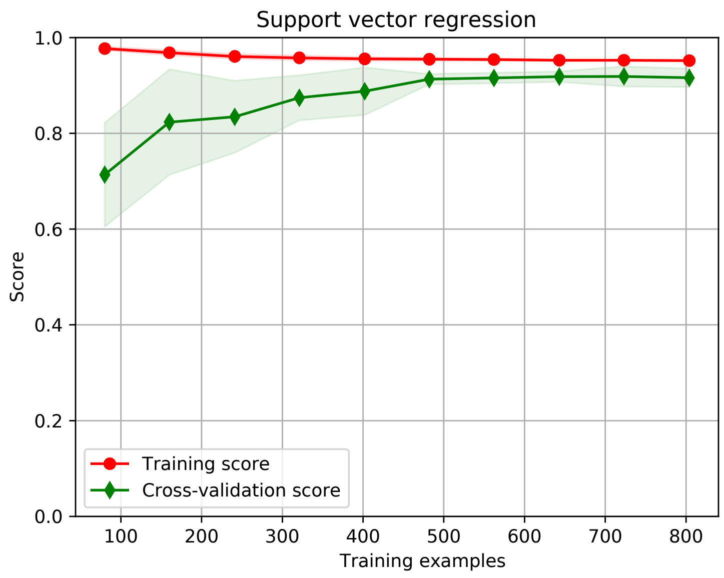


(c) Learning time of neural networks

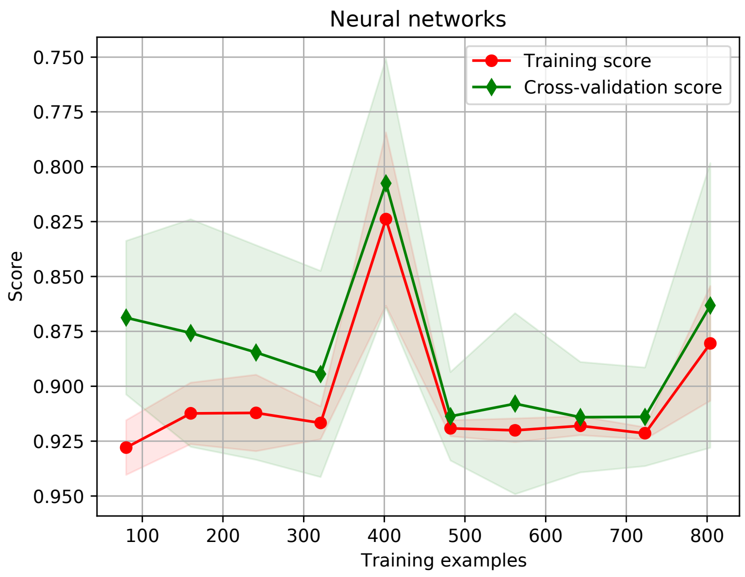
Figure 3.10: Learning time comparison of different machine learning algorithms incluing MLR, SVR and ANN



(a) Learning curve of multiple linear regression



(b) Learning curve of support vector regression



(c) Learning curve of neural networks

Figure 3.11: Selecting training model by using the learning curve of different machine learning algorithms including MLR, SVR and ANN

Table 3.4: Regression metrics evaluation of the models inlcuding MLR, SVR and ANN

Algorithm	**EV**	**MAE**	**MSE**	R^2
MLR	0.930009	0.337170	0.190819	0.930009
SVR	0.933951	0.323348	0.180074	0.933950
ANN	0.928588	0.341106	0.194697	0.928586

The Figures 3.10 and 3.11 show the learning time and learning curve of the three algorithms MLR, SVR, ANN. In Figures 3.11a, 3.11b, when the training size increases, the gap between training score and cross-validation score becomes smaller for both MLR and SVR, which means the loss errors converge to smaller values. However, the ANN model depends on the training size. When the training size is around 640, the model can reach optimal values.

Table 3.4 gives the comparison of Explained variation(EV), Mean absolute error (MAE), Mean squared error (MSE) and R^2 score for the three algorithms. SVR has the highest EV score and R^2 score compared with MLR and ANN. The MAE and MSE of SVR are also the lowest among the three algorithms. These results correspond to the validation results in Table 3.3 .

3.3 Performance Evaluation

3.3.1 Handover Trigger Phase

The traditional handover is mainly based on the RSSI value. When the STA goes out of the range of the associated AP and the RSSI value drops below the predefined threshold, the STA initiates the handover process. With the advancement of smartphones and other portable devices, the number of STAs for different purposes is growing rapidly. For example, if one STA application requires a higher data rate and another requires a lower data rate, it may be a good time to initiate handover for the application that requires a high data rate but maybe not good for the one that requires less data rate. However, handover triggered by RSSI cannot satisfy the data rate requirement.

Therefore, a handover trigger based on multiple network attributes for different application requirements is very necessary for 802.11 WLANs. In the machine learning based handover approach proposed, seven attributes including RSSI, SNR, packet delay, packet loss, data rate, average throughput and AP load are all considered to initiate handover.

As the neighbor list is orddered based on the AP score measured by the network attributes, the handover initiation is to compare the current serving AP score with the lowest AP score in the neighbor list. Let S_{cur} denote the serving AP score and T_{nei} denote the lowest AP score in the neighbor list. The lower score means better AP candidates. If $S_{cur} > T_{nei}$, the handover will be initiated. At the same time, the neighbor list will be updated according to the latest network measurements.

3.3.1.1 Handover Algorithm Description

The proposed handover algorithm for the neighbor list based on machine learning is summarized in Algorithm 6. The handover procedure is shown in Figure 3.12.

3.3.2 Handover Performance Evalutation

In this section, the handover performance of NLM is compared with the 802.11 standard and Neighbor Graph (NG) [3] which is a widely used method to establish neighborhood relationships. Four scenarios have been investigated including changing the speed of STAs, changing the number of APs, changing the number of STAs and changing the network topology. The network performance including throughput, packet loss rate and handover delay has been discussed in these four scenarios.

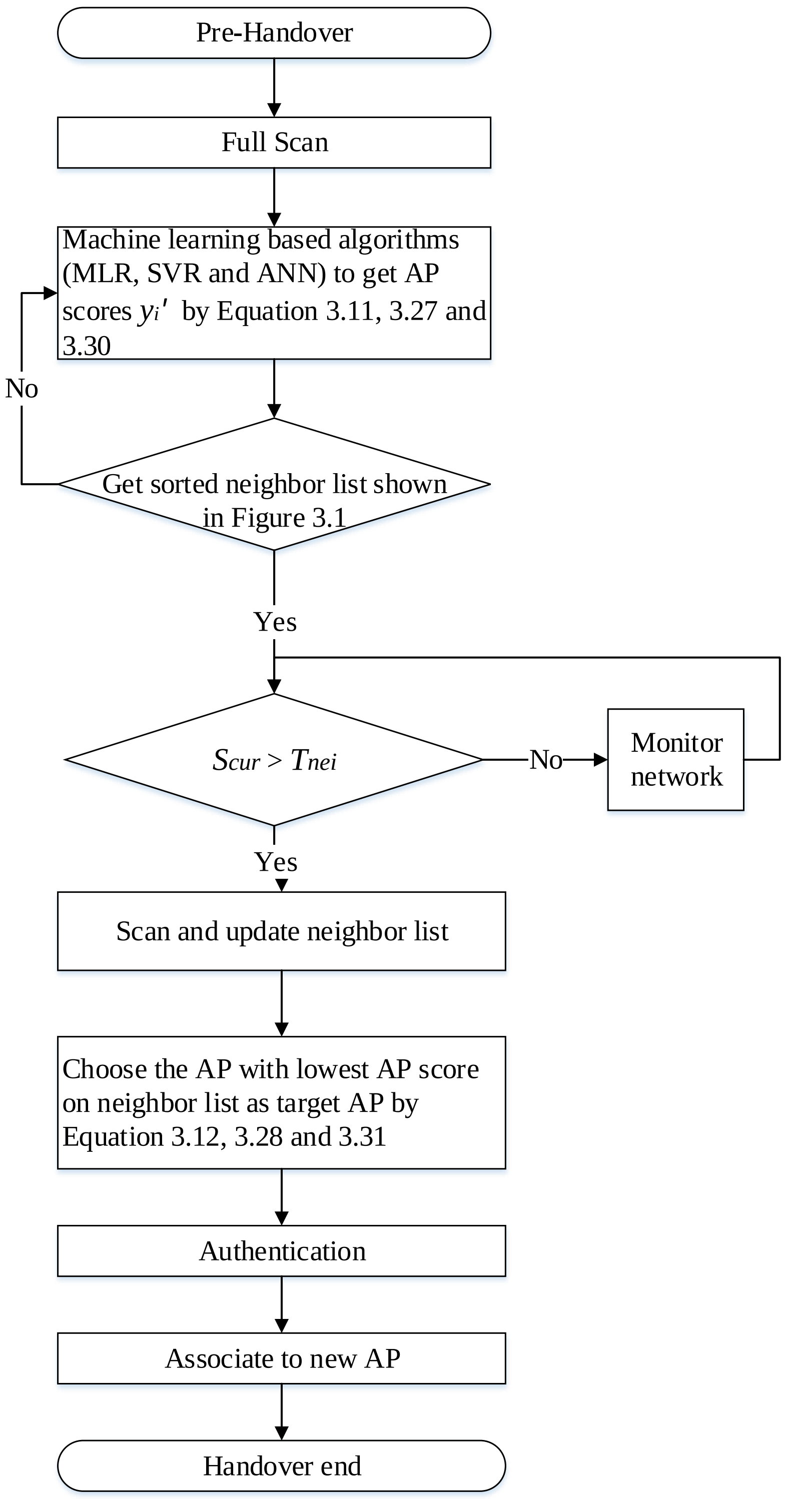

Pre-Handover
Full Scan
Machine learning based algorithms (MLR, SVR and ANN) to get AP scores y_i' by Equation 3.11, 3.27 and 3.30
No
Get sorted neighbor list shown in Figure 3.1
Yes
$S_{cur} > T_{nei}$
No
Monitor network
Yes
Scan and update neighbor list
Choose the AP with lowest AP score on neighbor list as target AP by Equation 3.12, 3.28 and 3.31
Authentication
Associate to new AP
Handover end

Figure 3.12: Machine learning based NLM handover procedures

Algorithm 6 Machine learning based neighbor list

```
1: Define simulation parameters and collect the scanning parameters by
   measurement report provided by 802.11;
2: Save parameters of all APs in the Neighbor AP List Matrix (NALM);
3: Train machine learning model (MLR, ANN, SVR) to get the value of
   metric for better handover performance of $AP_i$ named APSV (Access
   point score Value);
4: Save the APSV in NALM;
5: Rank the AP based on APSV value, ascending;
6: Monitor and compare the serving AP's APSV with the lowest AP's
   APSV in NALM;
7: If the serving AP's APSV is higher than the lowest APs APSV, han-
   dover will be performed.
```

3.3.3 Simulation Setup

The NLM optimization is implemented in NS-3 with each AP operating in the 2.412 GHz band (Channel 11) using the IEEE 802.11k protocol (802.11g PHY layer). Each AP has four Omni-directional antennas and deliberately limited to a maximum data rate of 54 Mbps (at the physical layer, 36.5 Mbps at the application layer). All results shown in this chapter are averages from 35 runs. Each run uses a different seed to generate a random data rate and packet size. The random traffic generation aims to satisfy different application requirements including voice call, video call, video streaming, online gaming in the different density network environments and network topologies. The averages shown are reported with a confidence interval of 95.00% under the assumption that the averages are normally distributed. This statistical methodology validates the simulation results. The simulation results are also validated by comparing proposed algorithms with conventional 802.11 standard and Neighbor Graph (NG) [3] using the same parameters. The simulation parameters are shown in Table 3.5.

Table 3.5: Simulation Parameters

Parameter	**Value**
Simulation time (t)	60s, 100s
Speed	2m/s - 6m/s
Number of AP	6 - 30
Distance between two APs	12 - 60m
Number of Mobile Nodes	1-35
ActiveProbing	true
WiFi Standard	802.11g, 802.11k
Packet size	10 - 10000 byte
Maximum data rate	54 Mbps
ProbeRequest Interval	0.20s
Channel Switching Interval	0.00025s
MaxChannelTime	0.15s
MinChannelTime	0.01s
SVR kernel	rbf
SVR γ	0.00066
SVR ϵ	0.1
ANN hidder layer	2
ANN activation	tanh

3.3.4 Analysis of Results

3.3.4.1 Changing Speed of STAs

In the first scenario (Scenario I), the handover performance is studied with increasing the speed of STAs. The speed of STAs takes a value between 2m/s to 6m/s within the distance of 360m. At the simulation start time $t = 0$s, the STAs are on the same location and are connected to the first AP. The simulation time is 60s for each run. The scenario is set up with six

APs and two STAs and the distance between each AP is 60m. The STAs start to move from AP0 to AP5 at t=0s with a constant speed. As the STAs move from AP0 to AP5, a handover occurs, but the handover performance is different when using different algorithms.

- **Throughput**

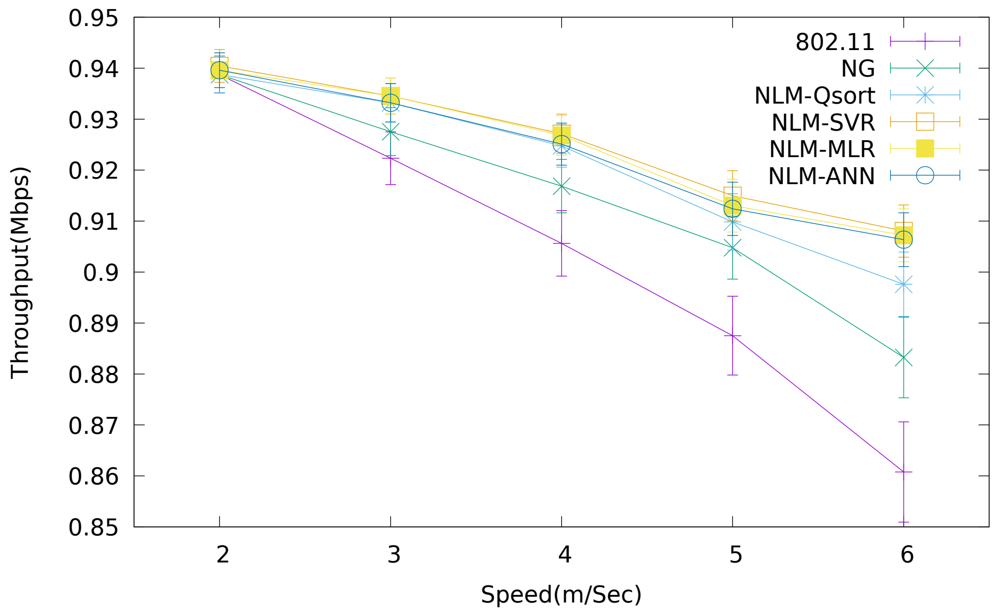


Figure 3.13: Throughput vs. Speed using different algorithms (Scenario I)

The average throughput has been compared when varying the speed of STAs to better understand the handover performance of NLM. The average throughput decreases as the moving speed increases in all mechanisms. This is because a faster STA causes a greater handover frequency. In Figure 3.13, the average throughput of IEEE 802.11 drops 8.30% when the speed changes from 2m/s to 6m/s. The average throughput has been improved by 1.24% using NG compared with the 802.11 standard. However, in the case of the proposed NLM handover scheme, throughput is less affected compared with the 802.11 standard and NG. The standard 802.11 scans all channels and NG generates neighborhood relationships based on more STAs and a long computation time. The NLM approach keeps the neighbor list updated for the STAs to associate with a new AP faster than 802.11 and NG by reducing the number of scanned channels and the time to establish neighborhood relationships.

In Figure 3.13, the average throughput has been improved by around 1.97% with NLM-Qsort, 2.44% with SVR, 2.35% with MLR and 2.25% with ANN. The NLM-Qsort algorithm is worse than machine learning algorithms. This is because the Quick-sort needs more time to update the neighbor list every scan. The long calculation time causes a long scan time. The machine learning algorithms without calculation time can get the updated neighbor list much faster than Quick-sort. The learning based NLM further reduces scanning delay compared with the NLM using Quick-sort.

Among the three machine learning algorithms, SVR shows the best performance. This is because SVR can avoid the problems such as overfitting which happens in linear regression. The overfitting problem can increase the prediction error of the AP score. If the AP score cannot be predicted correctly, the best target AP cannot be selected by the STAs. This can affect the throughput performance if the STAs handover to a wrong AP. Thus, the throughput performance of SVR is better than MLR. Also, ANN is a deep-learning algorithm so it uses more layers and hidden nodes to improve the learning accuracy. This deep learning algorithm needs more training time, so it cannot meet the real-time network requirements. The AP score predicted by SVR has high accuracy because SVR is without overfitting and has a fast training time using the small size of the support vectors [122]. Therefore, SVR easily satisfies the change of RSSI and communication quality caused by different speed movement and achieves the best improvement of throughput.

- **Packet loss rate**

  When handover occurs, it usually causes packets loss due to the momentary loss in connectivity.

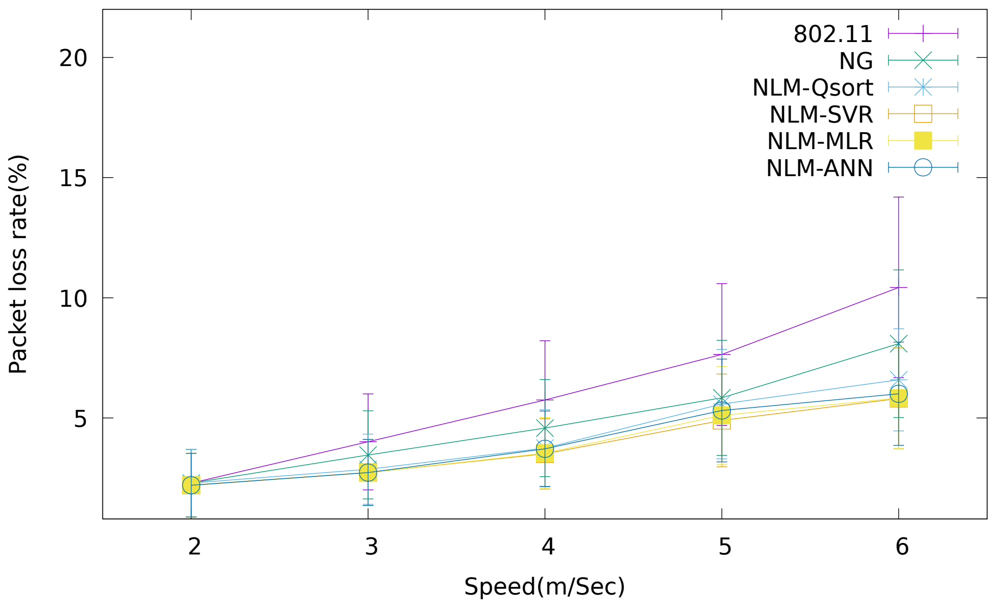


Figure 3.14: Packet loss rate using different algorithms (Scenario I)

Figure 3.14 illustrates the impact of handover on the data reception in different methods: 802.11, NG, NLM Q-sort and NLM using machine learning algorithms. The average packet loss rate for the 802.11 standard, NG, NLM-Q-sort, NLM-ANN, NLM-MLR, NLM-SVR are 6.48%, 4.85%, 4.07%, 3.99%, 3.88%, 3.83%, respectively. The average packet loss rate of NG, NLM-Q-sort, NLM-ANN, NLM-MLR, NLM-SVR compared with the standard 802.11 are reduced by 25.15%, 37.19%, 38.42%, 40.12%, 40.90% respectively.

When the speed of STAs increases from 2m/s to 6m/s, the communication time between the faster APs and STAs is reduced. This means the scanning process needs to be faster for a higher speed than for a lower speed for the STAs to find the best candidate AP successfully. In traditional 802.11 networks, the STAs always scan the same number of channels and APs. Thus, the scanning delay generates more packet loss for the faster STAs. In NG, the neighbor graph is established based on the reassociation request messages collected from STAs. The short communication time of high speed STAs cannot provide and update the neighbor information in time. However, the proposed neighbor list can be updated periodically based on dynamic network conditions. With the updated neighbor list, the STAs only scan the updated APs on the list and selects the best candidate AP, informed by the APC, based on the prediction by the NLM algorithms. Thus, the data disruption time is reduced by reducing the scanning delay and the packet loss rate is reduced because of the ongoing data transfer.

- **Handover delay**

  Figure 3.15 shows how the average handover delay is impacted by different algorithms with various speeds. For higher speed, the number of handovers certainly increased. However, compared with standard 802.11, the average handover numbers of the NLM using machine learning algorithms are all reduced to three.

  The 802.11 standard handover has an average handover delay of 0.20s/ho (handover number). The average handover delay is reduced by 26.63% (0.15s/ho) using NG, 33.62% (0.14s/ho) using NLM-Qsort, 35.28% (0.13s/ho) using NLM-ANN, 40.11% (0.12s/ho) using NLM-MLR and 55.09% (0.09s/ho) using NLM-SVR. The total handover number of machine learning based NLM is reduced to three. NG and NLM-Qsort are reduced to four as shown in Figure 3.15. Thus, the total handover delay of NLM-Qsort is higher than machine learning based NLM. As the learning-based NLM is based on the dynamic network requirements and includes more network parameters than NG and NLM-Qsort to update the neighbor list, the

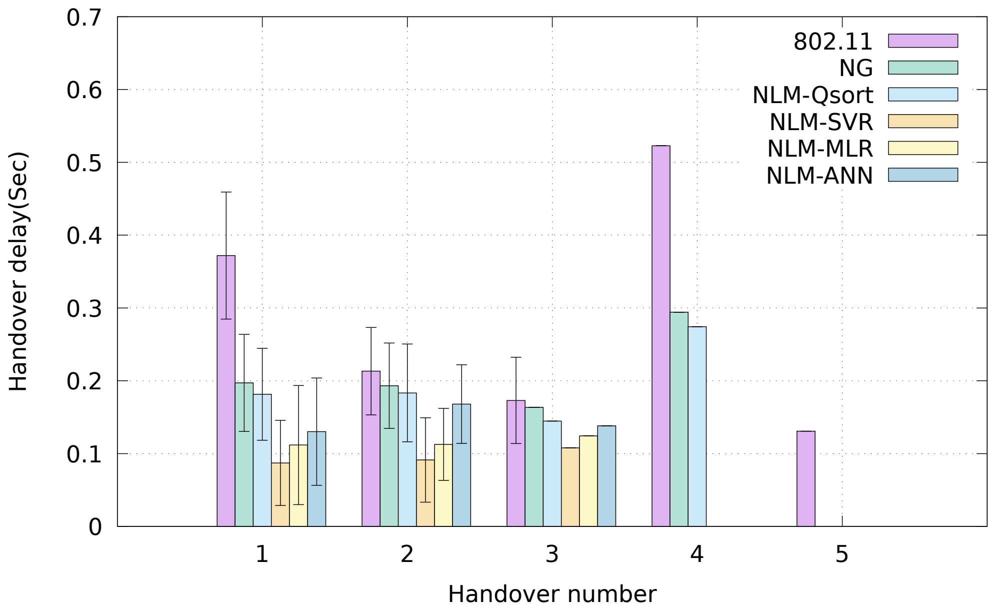


Figure 3.15: Handover delay vs. Handover number using different algorithms (Scenario I)

STAs can associate with a better AP than NG and NLM-Qsort. With faster STAs, unnecessary handover is avoided. Therefore, the total number of handovers is reduced.

3.3.4.2 Changing the Number of APs

In scenario II, the density of the network has been investigated by changing the number of APs. The STAs move between different numbers of APs (6, 12, 18, 24, 30) at a constant speed of 3m/s within the distance of 360m. The distance between each AP ranges from 12m to 60m. At the simulation start time $t = 0$, the STAs are at the same location and are connected to the first AP. The STAs start to move at t=0s from the first AP to the last AP with a constant speed. The simulation time is 60s for each run.

- **Throughput**

  With the increasing number of APs, the handover frequencies increase and the STAs have more chances to find new APs that can provide better throughput than the current serving AP.

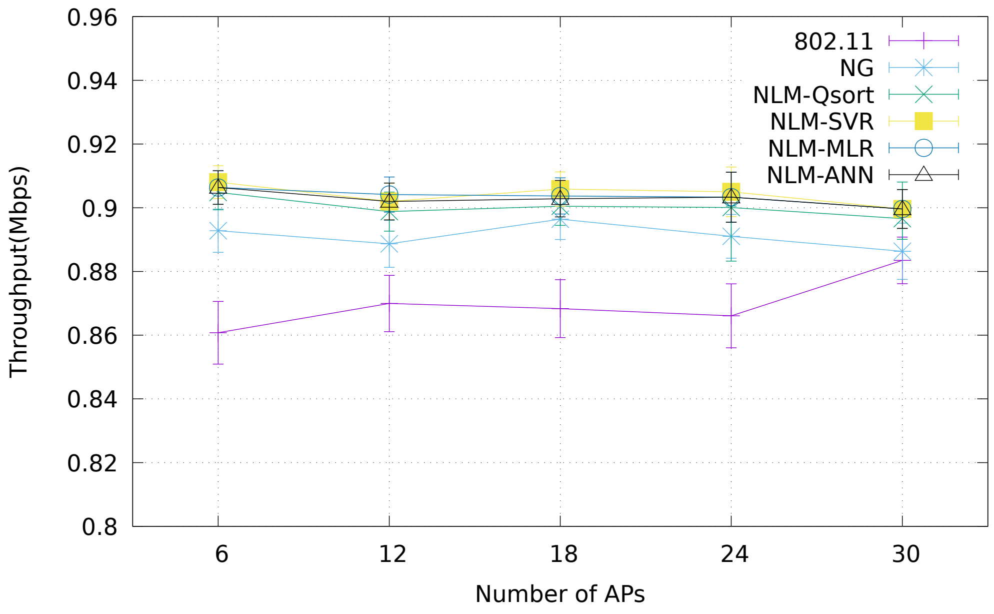


Figure 3.16: Throughput vs. Number of APs (Scenario II)

As can be seen in Figure 3.16, when the number of APs changes from 6 to 30, the throughput of standard 802.11 increases. This is because the STAs have more chances to find a new AP which can provide better throughput than the current serving AP. However, comparing the throughput when the number of APs is 12, 18 and 24, the observation is that increasing the number of APs does not always lead to increases in the throughput. The throughput becomes saturated. This is because the interference increases in dense networks even though the APs provide more and better candidate APs. Although the average throughput of NG is improved 2.93%, the throughput of NLM is less affected in the dense environment than NG when the number of APs increases. This is because NG does not consider the AP load and priority of APs when the neighbor graph is generated. The load imbalance and no-sorted AP causes a long scanning time to find the best candidate AP for STAs to reassociate with.

Figure 3.16 shows the average throughput of 802.11, NG and NLM approaches. The average throughput of the proposed NLM method has been improved by 3.98% with Q-sort, 4.43% with SVR, 4.12% with MLR and 4.29% with ANN compared with the 802.11 standard. As the learning-based neighbor list takes into consideration the density of AP, the average throughput and the interference between APs, the APs are sorted based on real-time network conditions. Thus, the throughput performance is less affected than 802.11 and NG in the dense network environment. The STAs are prevented from associating with an AP providing lower throughput. Therefore, the overall throughput performance of learning based NLM is better than 802.11 and NLM-Qsort.

- **Packet loss rate**

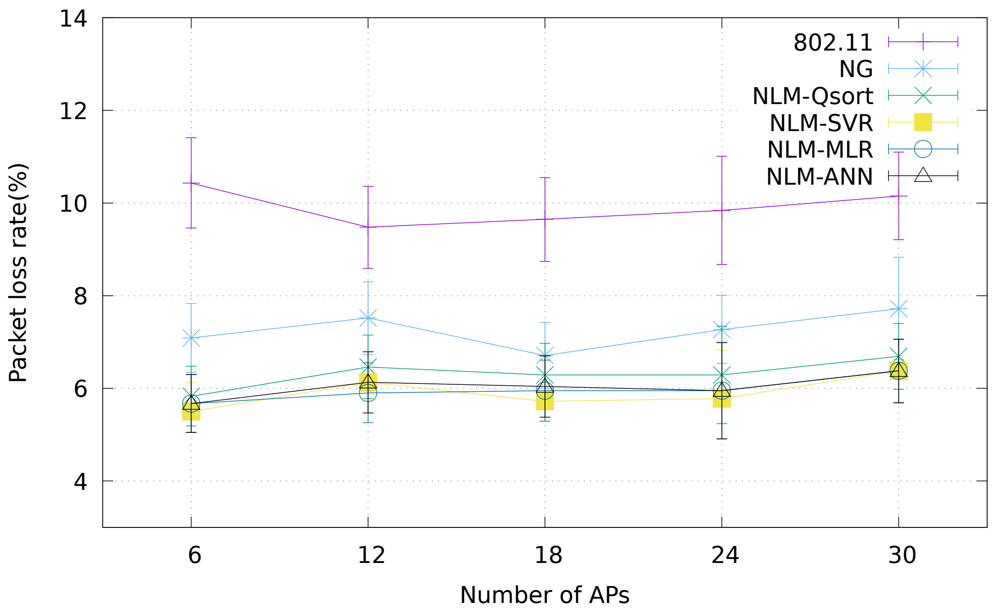


Figure 3.17: Packet loss rate vs. Number of APs (Scenario II)

Packet loss happens when handover occurs because of the disruption time. The packet loss rate also strongly depends on the density of a network. When the number of APs increases from 6 to 12, the trend of packet loss rate reduces because the STAs have more chances to associate with new APs that can provide better service than the current AP. However, as the number of APs increases, the packet loss rate is higher because of the increased interference between APs

The average packet loss rate of NG is reduced by 26.74% compared with 802.11. However, the average packet loss rate of NLM compared with 802.11 is reduced by 36.32% using Q-sort, 39.95% using SVR, 39.76% using MLR and 39.15% using ANN. The NLM algorithms consider the packet delay and packet loss parameters when the neighbor list is updated. The best candidate AP on the neighbor list is selected based on the dynamic network conditions. Thus, the STAs can handover to an AP that causes the lowest packet loss rate. With the NLM, the overall packet loss rate is reduced when compared with the 802.11 standard and NG.

- **Handover delay**

  Figure 3.18 shows how the average handover delay is impacted by different algorithms with various numbers of APs. The number of handovers increases because the density of APs increases. This involves more handover delay when handover occurs. However, compared with standard 802.11, NG and NLM-Qsort, the average handover delay of NLM using the machine learning algorithms is reduced.

  The average handover delay of 802.11 is 0.29s. The average handover delay of NG is reduced by 18.24% compared with 802.11. The NLM approaches compared with 802.11 reduce handover delay by 29.87% using Q-sort, 53.55% using MLR, 56.51% using SVR and 54.33% using ANN. As the dense network can increase the interference be-

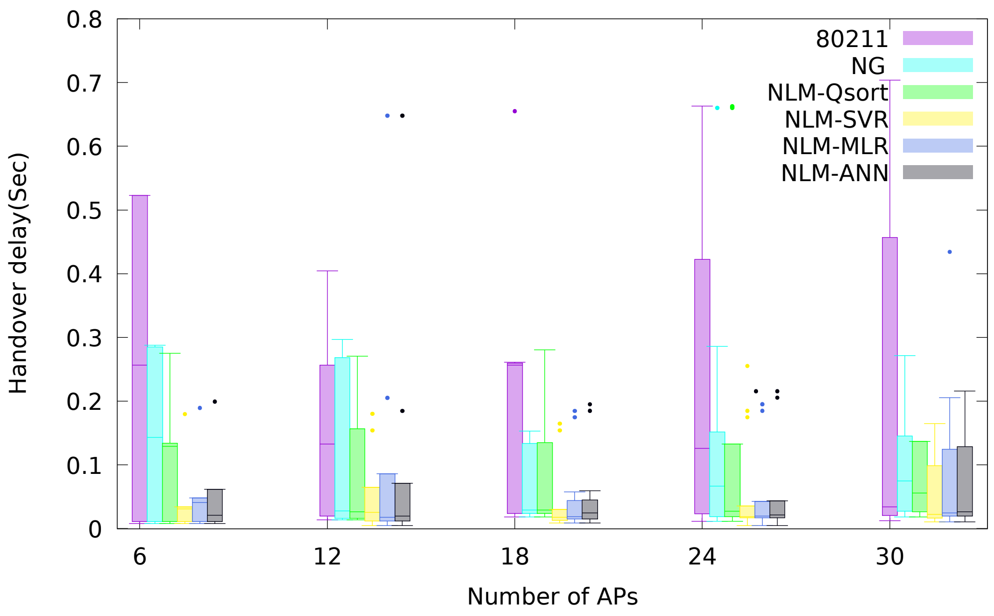


Figure 3.18: Handover delay vs. Number of APs (Scenario II)

tween APs, this interference forces the STAs to spend a longer time finding the best candidate AP. However, the learning-based NLM algorithms use multiple network parameters to sort the APs on the neighbor list. These network parameters including RSSI, SNR, delay and packet loss rate reflect the real-time network conditions. The STAs can select the best AP faster using the up-to-date neighbor list. Therefore, the handover delay is reduced. Overall, the handover performance of NLM algorithms shows better performance than standard 802.11 with the sorted neighbor APs and reduced channel numbers.

3.3.4.3 Changing the Number of STAs

In Scenario III, the number of STAs has been increased to see the impact on handover performance. The STAs move at t=0s with at a constant speed of 3m/s from AP0 to AP5. The distance between APs is 30m. The simulation time is 60s for each run.

- **Throughput**

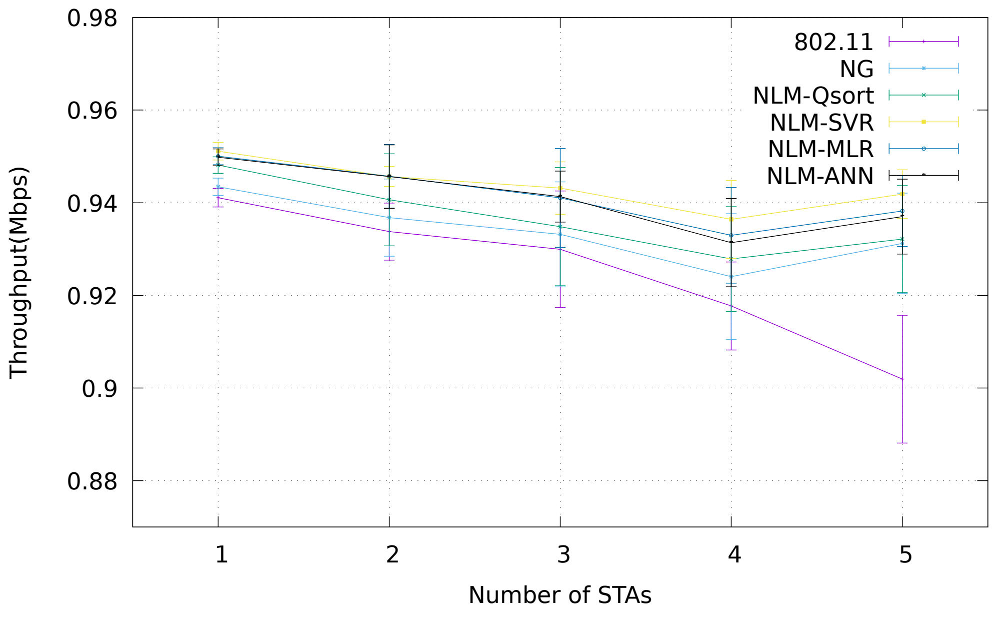


Figure 3.19: Throughput vs. Number of STAs (Scenario III)

As can be seen in Figure 3.19, the more STAs there are to utilize the capacity of APs, the lower the network throughput becomes. This is because all the STAs move at the same time and they all try to associate with the same AP. When the AP offered load is high, the imbalanced load leads to considerable degradation of throughput performance. In the traditional 802.11 standard, a STA selects an AP with a higher signal strength without considering the amount of load experienced by an AP. Compared with 802.11, although the average packet loss rate of NG has been reduced by around 16.02%, NG has not taken into account the overload issue in dense STA environment. However, NLM using machine learning algorithms updates the neighbor list not only based on the signal strength but also considers the AP load balancing. Therefore, the throughput is less affected by the STAs density.

Compared with the 802.11 standard, the average throughput has been improved by around 0.96% with NG, 1.20% with NLM-Qsort, 2.03% with NLM-SVR , 1.81% with NLM-MLR and 1.75% with NLM-ANN. The improvement of machine learning based NLM is better than NG and NLM-Qsort. This is because the learning based neighbor list is updated considering average throughput and AP load metrics based on real-time networks conditions. With the updated neighbor list, the STAs can be prevented from reassociating with an AP with degraded throughput and imbalanced load.

- **Packet loss rate**

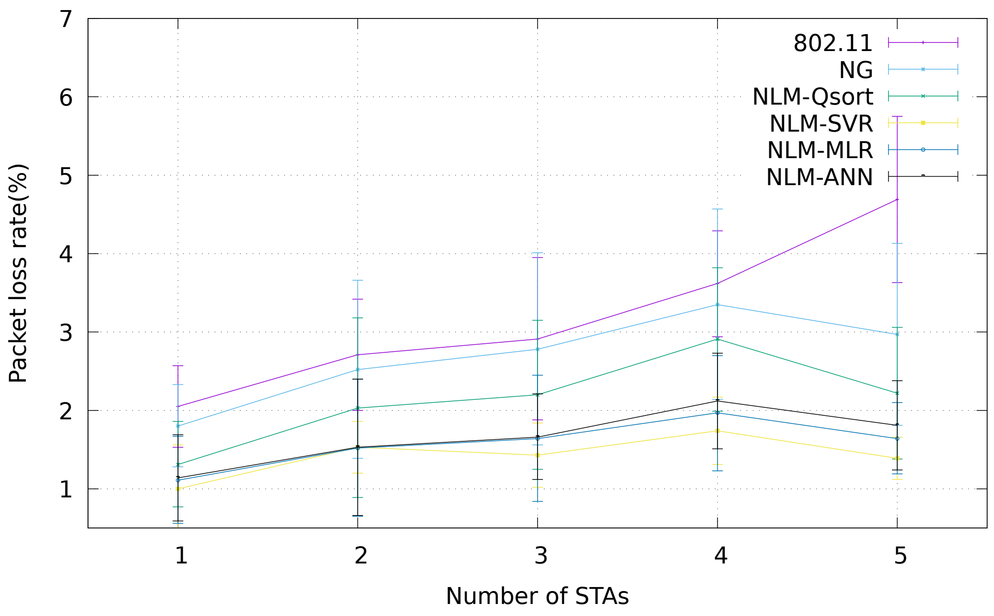


Figure 3.20: Packet loss rate vs. Number of STAs (Scenario III)

Figure 3.20 illustrates the average packet loss rate changes when the number of STAs increases. The packet loss rate increases when the number of STAs increases. This is because the larger the number of STAs is, the more contention there is among them. Thus, the larger

STA density causes a high packet loss rate. Also, if the AP load is not balanced, an overloaded AP cannot provide better performance even with strong signal strength. This leads to the STAs spending more time searching for a new AP. Therefore, the packet loss rate increases when using the 802.11 standard. As the NG approach has overload issue, the packet loss rate only reduces by around 18.37% compared with 802.11. However, the packet loss rate has been reduced by the NLM approaches with the updated neighbor list. The updated neighbor list with the reduction of APs can reduce the scanning time. Also, the machine learning-based algorithms of NLM-SVR, NLM-MLR and NLM-ANN sort the neighbor list considering the AP load conditions. Thus, compared with 802.11, the average packet loss rate has been reduced by around 33.23% using NLM-Qsort, 55.63% with NLM-SVR, 50.69% with NLM-MLR and 52.25% with NLM-ANN.

- **Handover delay**

  Figure 3.21 shows the average handover delay impacted by different algorithms by varying the number of STAs. As the number of STAs increases, the average handover delay increases because of the density of STAs in 802.11. In this scenario, the STAs move at the same time and try to associate with the same AP. This leads to a load unbalanced and causes the STAs to spend more time finding a new AP. However, compared with standard 802.11, the average handover delay of NLM using machine learning algorithms is not affected by the number of different STAs because it considers the AP load balance.

  Alghouth the NG approach reduces handover delay by 38.43%, the NLM algorithms further reduce the handover delay. The NLM approaches reduced handover delay by 41.71% using Q-sort, 51.97% using ANN, 52.03% using MLR and 54.09% using SVR, respectively. As the three machine learning based algorithms consider the AP load conditions, it avoids potentially poor handover decision that may overload the APs. Thus, the handover delay and unsuccessful handover are reduced compared with Q-sort.

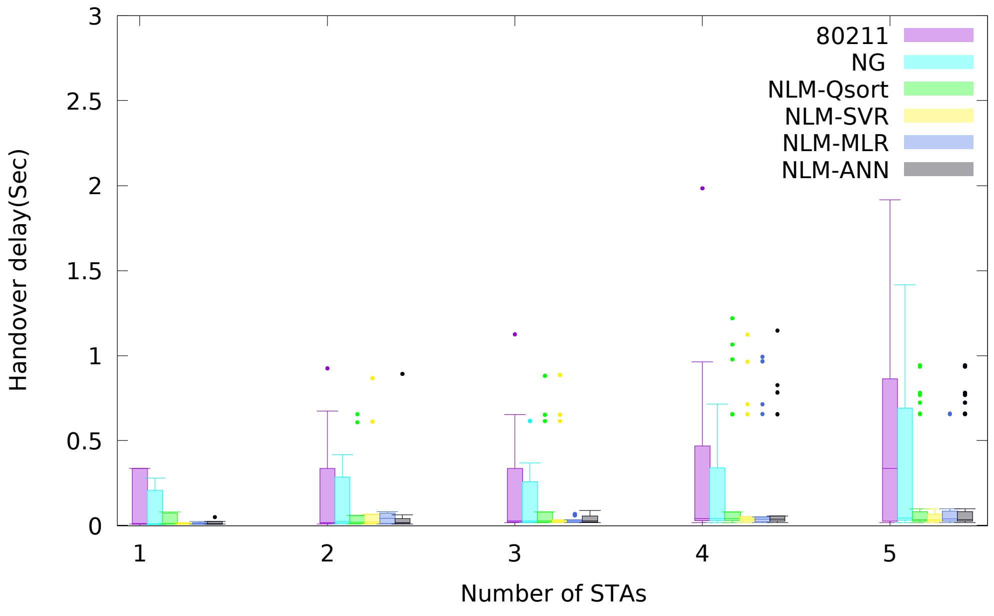


Figure 3.21: Handover delay vs. Number of STAs (Scenario III)

3.3.4.4 Changing the Network Topologies

In scenario IV, different topologies of AP placement are investigated. First, the APs are placed in one line as shown in Figure 3.22. The distance between any two APs is 30m. One STA moves from AP0 to AP5 at t=0s with a constant speed of 3m/s from left to right backwards and forwards three times. The simulation time is 100s for each run.

Then, the APs are placed in a grid as shown in Figure 3.23. The distance between the two lines is 16m and 4 APs placed on each line. At t=0s, one STA moves in the middle among lines back and forth with a constant speed of 3m/s. The distance between each row of APs is 60m.

Finally, Figure 3.24 shows the random AP placement (Poisson point distribution) in a 90×80m full coverage area. One STA moves with a con-

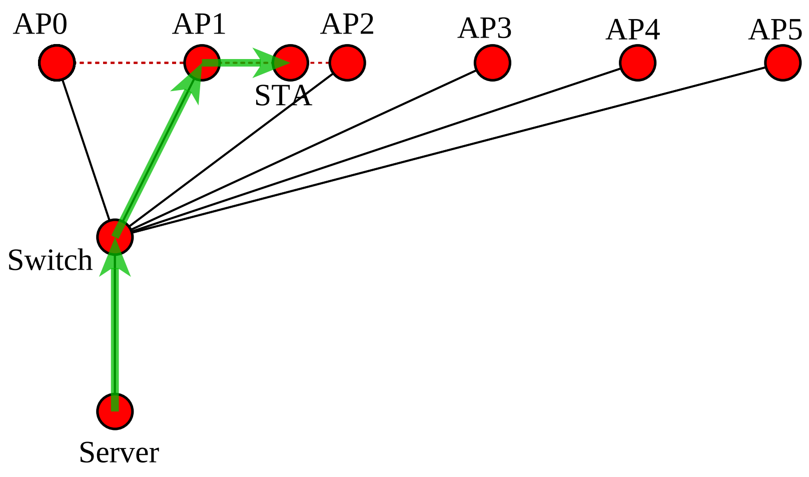


Figure 3.22: AP placement of 1D line topology in NS3

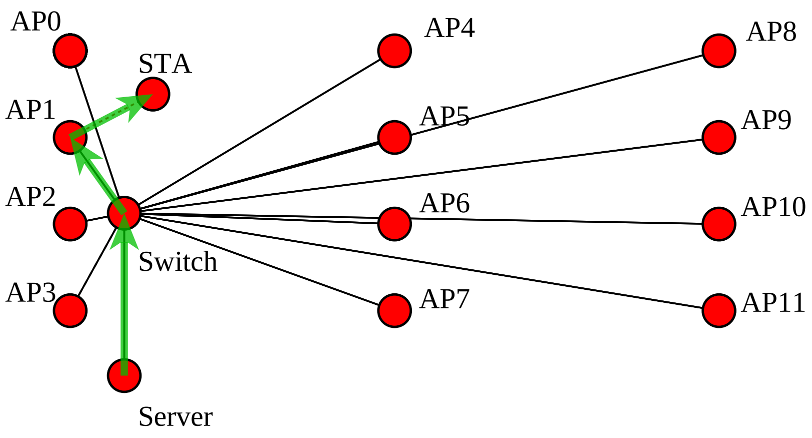


Figure 3.23: AP placement of 2D grid topology in NS3

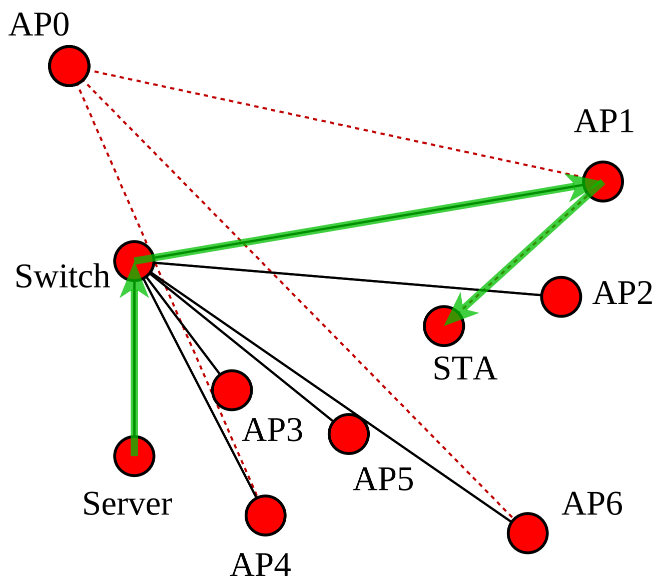


Figure 3.24: AP placement of random topology in NS3

stant speed of 3m/s from left to right backward and forwards for the full width of the coverage area.

- **Throughput**

  In Figures 3.25, 3.26 and 3.27, the average throughput of the NLM approaches and the standard 802.11 has been compared in three different network topologies. Figure 3.25 shows the average throughput changing when STA moves in a line netwoq1rk topology. Compared with 802.11, the average throughput of the proposed NLM method has been improved by 1.00% with NLM-Qsort, 1.28% with NLM-SVR, 1.18% with MLR and 1.08% with NLM-ANN, respectively. The average throughput of the proposed NLM method has been improved by 0.45% with NG. The throughput of line topology is not affected a lot by using different NLM algorithms. This line topology is not dense, and the STA moves with a constant speed. Thus, the STA can easily find the best candidate AP to handover.

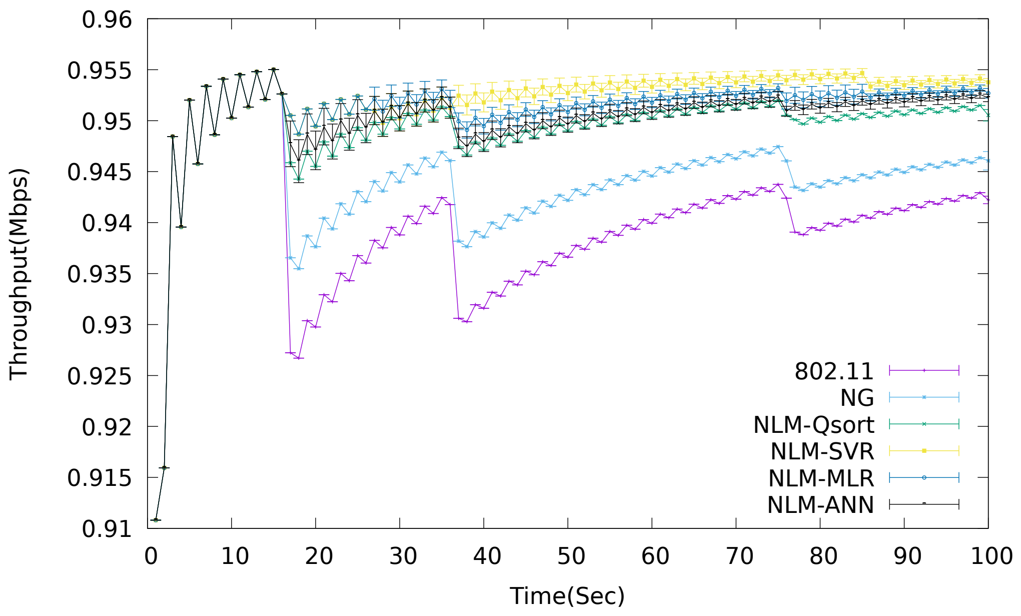


Figure 3.25: Throughput vs. Simulation time -1D topology (Scenario IV)

As SVR can predict with a high fitting accuracy in the simple topology networks, SVR achieves the best performance. ANN is a deep learning algorithm usually needs a large dataset with longer training time to improve accuracy. This can cause longer scanning delay than SVR. Figure 3.26 shows the average throughput when the STA moves in a grid network topology. The average throughput of the proposed NLM method has been improved by 4.13% with NG. Compared with 802.11, the average throughput of the proposed NLM method has been improved by 4.53% with NLM-Qsort, 5.41% with NLM-SVR, 5.37% with MLR and 5.28% with NLM-ANN. Q-sort shows the worst improvement in the grid network topology. This is because in the dense network, the number of APs is increasing and Q-sort needs more time to update the neighbor list. However, SVR, MLR and ANN can sort the neighbor list faster than Q-sort because of less learning time.

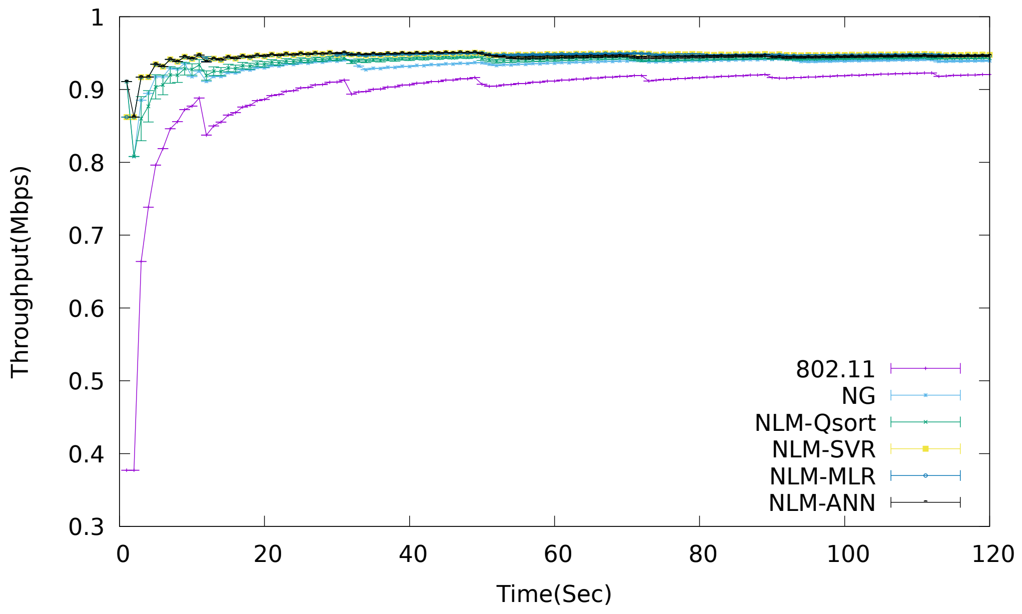


Figure 3.26: Throughput vs. Simulation time - 2D topology (Scenario IV)

Figure 3.27 shows the average throughput changing when the STA moves in a random network topology. The average throughput of the proposed NLM method has been improved by 2.58% with NG. Compared with 802.11, the average throughput of the proposed NLM method has been improved by 3.47% with NLM-Qsort, 4.20% with NLM-SVR, 5.37% with MLR and 4.34% with NLM-ANN. SVR shows the best improvement for the random network topology. This is because in the dense network, the number of APs is increasing and SVR can update the neighbor list with fast updating time. The three learning algorithms including SVR, MLR and ANN update the neighbor list considering the interference between APs, average throughput and packet delay. Thus, machine learning based NLM shows better performance than NLM-Qsort.

Compared with the throughput improvement in these three different topologies, the results show that 1D topology has less impact on

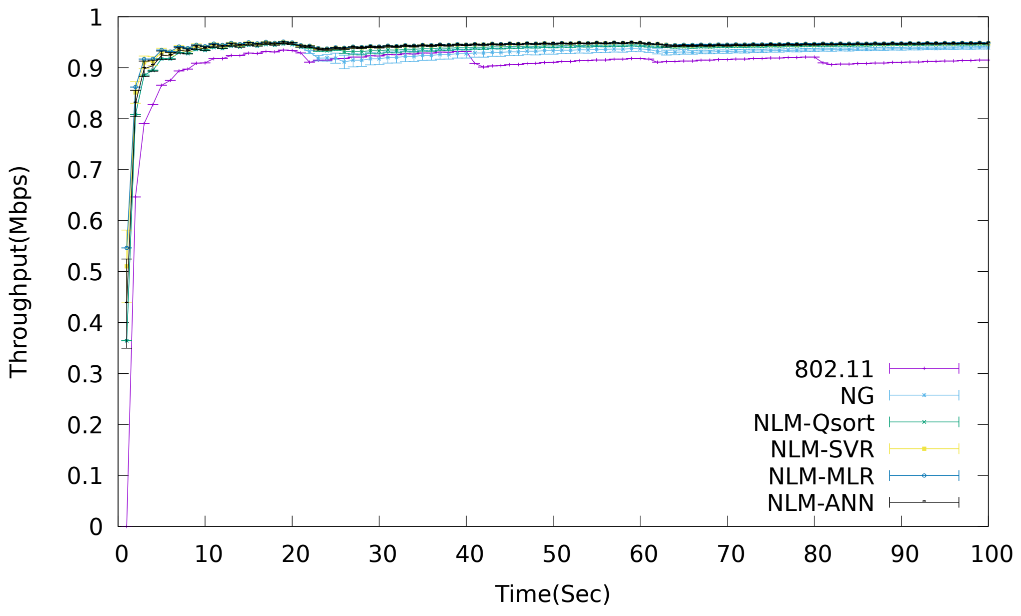


Figure 3.27: Throughput vs. Simulation time - Random topology (Scenario IV)

throughput than 2D and random topologies. This is because the AP density is different in different topologies. The different AP density and placement affect the STA's scanning time to find a new AP. In the 2D Scenario, the number of APs is larger than in the 1D and random topologies. Therefore, the NLM improvement is more obvious than the other two topologies because NLM reduces the number of scanning channels and APs. This reduces the total scanning time to improve throughput performance.

- **Packet loss rate**

  In Figures 3.28, 3.29 and 3.30, the packet loss rate of NLM approaches is compared with the standard 802.11 in three network topologies.

  Figure 3.28 shows the average packet loss rate changing when STA moves in a line network topology. Compared with 802.11, the average throughput of the proposed NLM method has been improved

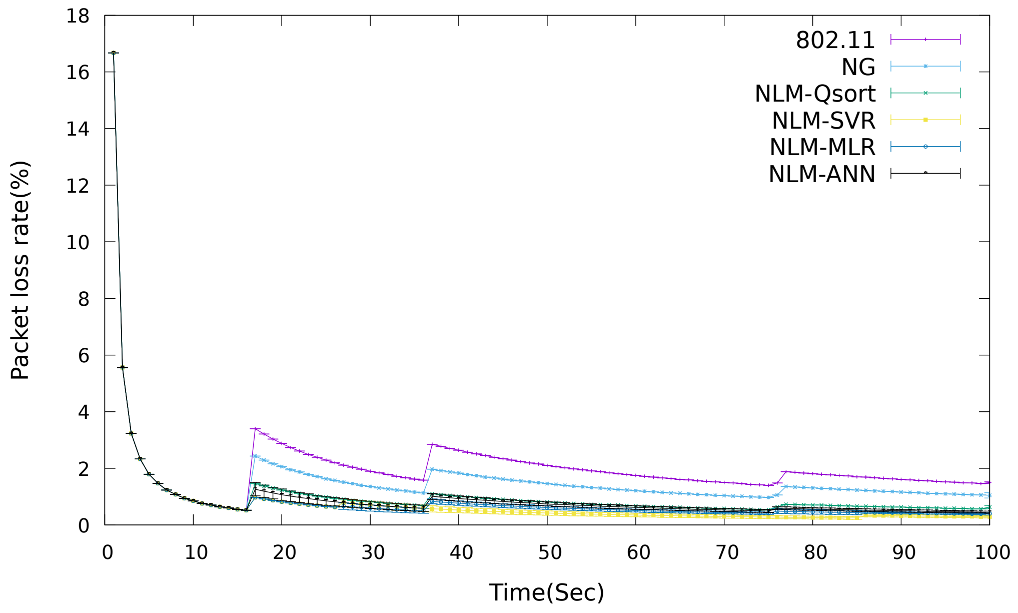


Figure 3.28: Packet loss rate vs. Simulation time - 1D topology (Scenario IV)

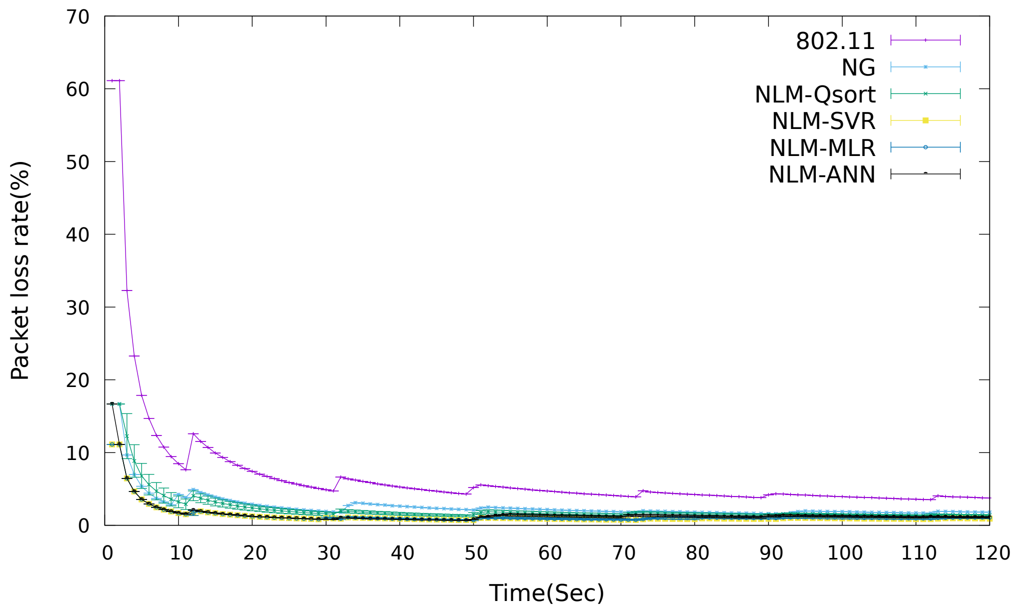


Figure 3.29: Packet loss rate vs. Simulation time - 2D topology (Scenario IV)

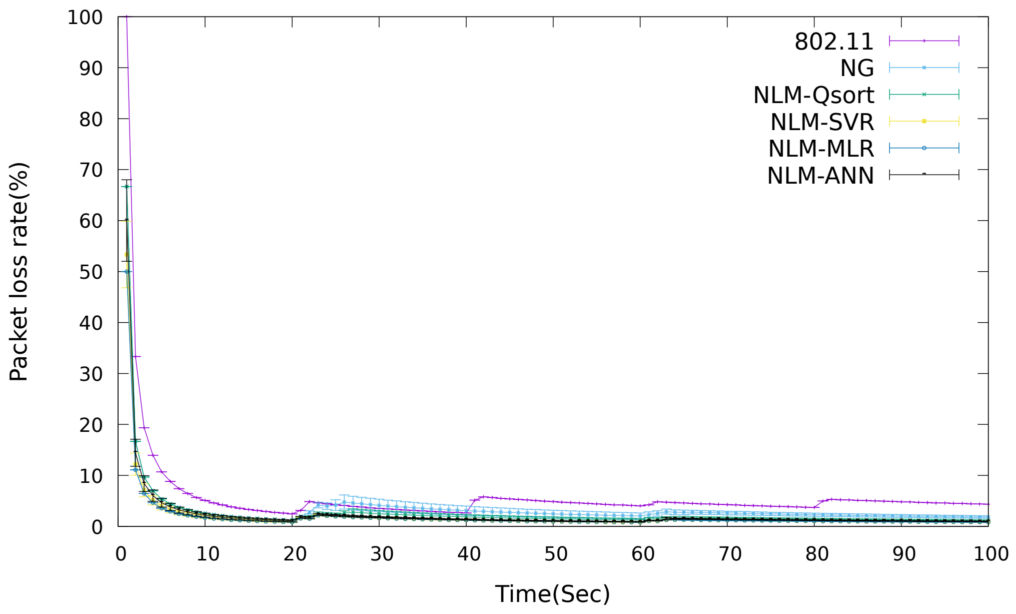


Figure 3.30: Packet loss rate vs. Simulation time - Random topology (Scenario IV)

by 48.51% with NLM-Qsort, 62.37% with NLM-SVR, 57.43% with MLR and 52.48% with NLM-ANN. The average throughput of the proposed NLM method has been improved by 32.67% with NG.

Figure 3.29 shows the average packet loss rate when STA moves in a grid network topology. Compared with 802.11, the average throughput of the proposed NLM method has been improved by 65.17% with NLM-Qsort, 78.01% with NLM-SVR, 77.40% with MLR and 76.63% with NLM-ANN. The average throughput of the proposed NLM method has been improved by 59.13% with NG.

Figure 3.30 shows the average packet loss rate changing when STA moves in a random network topology. Compared with 802.11, the average throughput of proposed NLM method has been improved by 54.39% with NLM-Qsort, 65.70% with NLM-SVR, 67.74% with MLR and 64.34% with NLM-ANN. The average throughput of the proposed NLM method has been improved by 39.86% with NG.

As Q-sort does not consider the effect of packet loss rate effect when the neighbor list is updated, its effect on the packet loss rate is less than the machine learning-based algorithms. As discussed in section 3.2.2.3, SVR can learn fast to find the global optimal solution with high fitting accuracy. Thus, SVR performs the best to solve the non-linear neighbor list updating problem. While ANN needs more learning time than SVR, the improvement is less than SVR but still higher than MLR which is more suitable for solving the linear issue.

- **Handover delay**

Figure 3.31 illustrates the handover delay comparison among three different topologies. In the 2D topology, as the density of APs is higher than 1D and random topology, the handover occurs more frequently than the other two topologies. Therefore, the average handover delay of 2D is larger than the 1D and random topologies in 802.11. However, the NLM approaches reduce all the handover delay for the three topologies. The NLM approaches have less effect on handover performance when the network topologies are different. This is because the NLM dynamically updates the neighbor list based on the real-time network environment. The best AP on the list can be updated timely when STA starts a new scan. Compared with the learning algorithms, NG and Q-sort achieve less improvement than SVR, MLR or ANN. This is because NG has overload issues and Q-sort has to calculate the sorting algorithm every time for updating the neighbor list and the sorting time is much longer than the learning time. This cannot satisfy the dynamic network requirements. The NLM-SVR still shows the best performance because it can update neighbor list with the highest accuracy and fastest learning time for solving the non-linear problem.

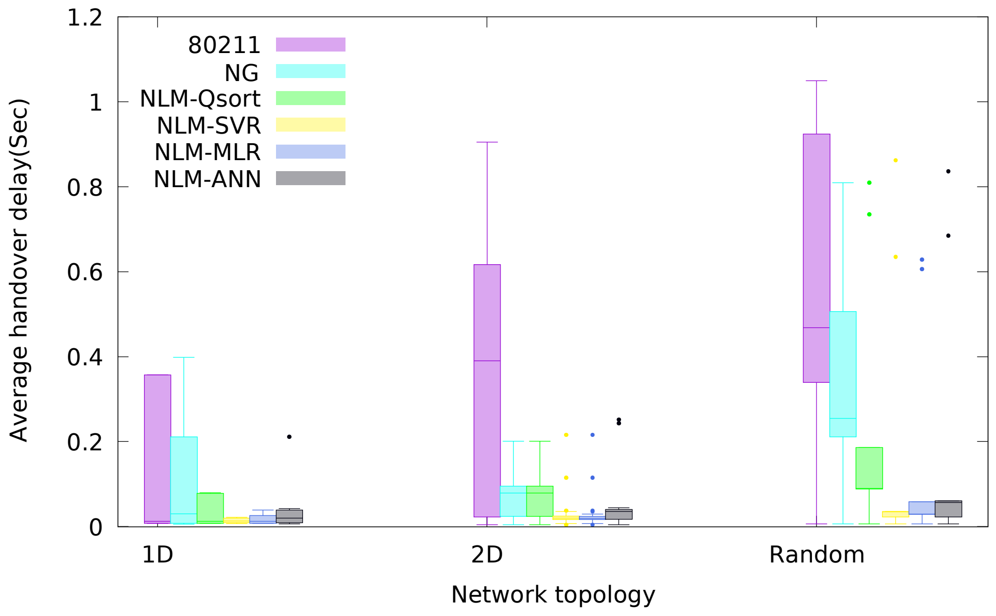


Figure 3.31: Handover delay in three different topologies (Scenario IV)

3.4 Summary

The proposed neighbor list mechanism reduces handover delay to improve mobility QoS for WiFi networks. The neighbor list can be updated by the ranking algorithms including Q-sort or the three machine learning algorithms including SVR, MLR and ANN. Compared with the standard 802.11 and NG, the NLM-Qsort has enhanced the handover performance because of the reduction of scanning channels and APs. Moreover, the machine learning algorithms consider more network features such as RSSI, delay and AP load to adapt to the dynamic network environment. Therefore, the performance has been further improved by machine learning-based algorithms. However, the scanning timers that can also affect the scanning delay are not considered in NLM. Thus, the next chapter will introduce a self-optimization approach to further improve handover performance.

The proposed NLM is stored in the AP controller. In the future, the neighbor list will be updated on the mobile host in order to reduce the link latency between the controller, AP and STAs.

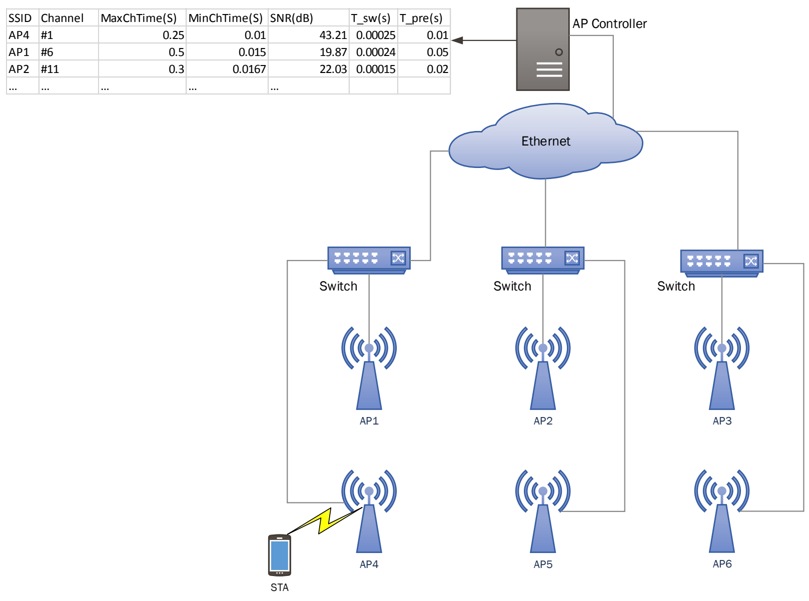
Chapter 4

Self-optimizing Scanning Parameters

The Neighbor List Mechanism (NLM) discussed in Chapter 3 has reduced handover delay by reducing the number of channels and APs scanned by STAs. However, the scanning parameters in the NLM are based on the default fixed values. In this chapter, a Self-Organizing Network (SON) inspired approach using self-optimizing scanning parameters optimized by a Genetic Algorithm(GA) is proposed. These self-optimizing adaptive parameters are dynamically adjusted based on real-time network conditions to reduce MAC layer handover delay. Each STA is made aware of the scanning parameters of each channel by a centralized controller. Therefore, the STA probes a set of channels and APs with adaptive parameters instead of fixed values. The proposed algorithms consider *all* the parameters in the scanning phase including *MinChannelTime*, *MaxChannelTime*, channel switching time, probe request interval, the number of channels scanned and the channel sequence. The scanning delay is reduced by reducing the unnecessary waiting time of probe responses of non-adjacent APs or APs faulty APs.

In the following sections, the proposed self-optimizing scheme will be firstly introduced and then give the problem formulation. Furthermore, how the scanning parameters are optimized by GA is explained. Finally, GA performance and handover performance are evaluated and analyzed.

4.1 Proposed Self-optimizing Scheme



SSID	Channel	MaxChTime(S)	MinChTime(S)	SNR(dB)	T_sw(s)	T_pre(s)
AP4	#1	0.25	0.01	43.21	0.00025	0.01
AP1	#6	0.5	0.015	19.87	0.00024	0.05
AP2	#11	0.3	0.0167	22.03	0.00015	0.02
...	...	...	...	...		

Figure 4.1: Self-optimizing network architecture including an AP controller (APC) and the table sent by the APC containing scanning parameters

In this section, the self-optimizing scheme of optimizing scanning parameters is introduced. The proposed self-optimizing approach uses a Genetic Algorithm (GA) to optimize scanning parameters and configure the scanning parameters for each STA before the scan process is initiated. Once the GA finds the optimal initial values of the scanning parameters, the table of GA optimized scanning parameters and channels are stored in the AP controller (APC). The GA parameters are refreshed when STAs start a new scan. Hence, GA parameters are unlikely to be outdated. Moreover, scanning timers can be further adjusted by the adaptive timers. The motivation behind using GA is to find the best candidate AP by scanning an up-to-date list instead of scanning all available channels by a STA. The up-to-date list with GA optimized timers for each channel is maintained in the APC.

The network scenario used is shown in Figure 4.1. It depicts a centralized controller connected to all the APs. Once the STA detects that the current associated AP's signal strength is lower than the predefined threshold, the active scanning is initiated. The tables of GA optimized scanning parameters and channels are sent to the STA by the controller, and then the STA starts to search for the best candidate AP using the optimized parameters.

4.2 Problem Formulation

The total handover time includes scanning time, re-association time and re-authentication time [8]. Therefore, the handover delay for a handover in 802.11 WLANs can be calculated by Equation 4.1,

$$\sum_{i=1}^{N} T_{Handover(i)} = \sum_{i=1}^{N} \{T_{sc(i)} + T_{au(i)} + T_{as(i)}\}, \tag{4.1}$$

where $T_{Handover(i)}$ is the handover delay of each channel i; $T_{sc(i)}$, $T_{as(i)}$ and $T_{au(i)}$ are scanning delay, association delay and authentication delay of each channel, respectively.

Self-optimizing aims to reduce the scanning delay because the scanning delay is accounting for 90% of the overall handover delay [26]. In the active scan, the scan time includes the probe time that is the time used to broadcast probe messages on the i-th channel, the switching time that is the time used by the STA to switch from current channel to the next channel and the waiting time that is the time used for collecting the response messages [8]. The scan delay can be calculated using Equation 4.2,

$$\begin{aligned}\sum_{i=1}^{N} T_{sc}(i) &= \sum_{i=1}^{N}\{T_{P_{req}}(i) + T_{wt}(i) + T_{sw}(i)\} \\ &= \sum_{i=1}^{N}\{T_{P_{req}}(i) + p_e(i) * t_{min}(i)) + (1 - p_e(i)) * t_{max}(i) \quad + T_{sw}(i)\},\end{aligned} \tag{4.2}$$

where the number of channels is denoted by N and the probability of the i-th channel is empty (given by $p_e(i)$); $T_{P_{req}}(i)$ denotes probe request time on channel i; $T_{sw}(i)$ denotes the switching time between channels; $T_{wt}(i)$ denotes the probe response waiting time on channel i; $MinChannelTime$ of i-th channel is denoted as $t_{min}(i)$ and $MaxChannelTime$ of i_{th} channel is denoted as $t_{max}(i)$.

If the probe time is longer on each channel, the probability of discovered AP will be higher. Therefore, the scanning process is assumed as an exponential distribution, the probability density function(PDF) as below:

$$f(t) = \lambda e^{-\lambda t}, \tag{4.3}$$

where λ is the channel load, t is the probing time. The probability of AP discovery within $t_{min}(i)$ of each channel can be calculated by Equation 4.4:

$$P_{min}(0 \le t \le t_{min}(i)) = \int_{0}^{t_{min}(i)} \lambda e^{-\lambda t} dt = 1 - \lambda e^{-\lambda t_{min}(i)}. \tag{4.4}$$

Therefore, the average discovery number of APs within $t_{min}(i)$ is calculated by Equation 4.5:

$$N_{Min} = P_{min} * N_{ap}, \tag{4.5}$$

where N_{ap} is the discovery number of APs over each scanning time $t_{max}(i)$ on channel i.

The average discovery rate of APs over a duration of $t_{max}(i)$ is expressed as follows:

$$\begin{aligned} D &= \frac{N_{Min}}{t_{min}(i)} + \frac{N_{ap} - N_{Min}}{t_{max}(i) - t_{min}(i)} \\ &= \frac{P_{min} * N_{ap}}{t_{min}(i)} + \frac{(1 - P_{min}) * N_{ap}}{t_{max}(i) - t_{min}(i)}. \end{aligned} \tag{4.6}$$

Let $A_i(ap)$ denote a set of available APs on channel i where $1 \leq i \leq n$ and $i \in \mathbb{Z}^+$ and S_i denote the set of Signal-to-Noise (SNR) value of each AP on channel i. The set of $A_i(ap)$and S_i are expressed as follows:

$$A_i(ap) = \{ap_{i,1}, ..ap_{i,k}, ...ap_{i,n}\}, ap_{i,k} \in \mathbb{N}, \tag{4.7}$$

$$S_i = \{snr_{i,1}, ..snr_{i,k}, ...snr_{i,n}\}, snr_{i,k} \in \mathbb{R}, \tag{4.8}$$

where $ap_{i,k}$ denotes the AP of channel i with index k and $snr_{i,k}$ denotes the SNR value of $ap_{i,k}$, $\mathbb{N}$ is the natural numbers $\{0, 1, 2, ...\}$ and $\mathbb{R}$ is the real number. Therefore, the highest SNR on channel i is expressed as follows:

$$SNR_h(i) = \max\{snr_{i,1}, ..snr_{i,k}, ...snr_{i,n}\}, snr_{i,k} \in \mathbb{R}, \tag{4.9}$$

where $SNR_h(i)$ denotes the highest SNR value in the set of S_i, and the sum of the highest SNR value for all the scanned channels can be written as $\sum_{i=1}^{N} SNR_h(i)$ where N is the number of channels scanned. Intuitively, a higher value of $\sum_{i=1}^{N} SNR_h(i)$ means that the number of APs discovered is high for this scan because the sum of $\sum_{i=1}^{N} SNR_h(i)$ is proportional to the discovery rate of APs (i.e. $\sum_{i=1}^{N} SNR_h(i) \propto D$). Therefore, for each channel i, the problem is formulated as follows:

$$\arg_{\underbrace{\{T_{P_{req}}(i), T_{sw}(i), t_{min}(i), t_{max}(i)\}}_{\text{chromosome}}} \max \frac{\sum_{i=1}^{N} SNR_h(i)}{\sum_{i=1}^{N} T_{sc}(i)} \tag{4.10}$$

subject to :

$$0 \leq T_{P_{req}}(i) \leq 200ms, 0 \leq T_{sw}(i) \leq 0.5ms, \tag{4.11}$$

$$5 \leq t_{min}(i)) \leq 15ms, 3 \leq t_{max}(i) \leq 90ms, \tag{4.12}$$

$$t_{min}(i) \leq t_{max}(i), \tag{4.13}$$

where the constraints expressed in Equation 4.11 - 4.13 are the boundary values for the scanning timers as used in [29].

The self-optimizing approach aims to search for the maximum value of the objective Equation 4.10 subject to the constraints in Equation 4.11-4.13. Also, the solutions of scanning parameters are in a finite and discrete set of objects to satisfy Equation 4.10 with conditions. The scanning parameters have interactions between each other to affect the scanning delay. For example, if the $t_{min}(i))$ is too long and STA finds available APs on the scanning channel, the STA has to spend $t_{max}(i))$ waiting time to search for more APs. The probe interval $T_{P_{req}}(i)$ also affects the $t_{min}(i))$ waiting time for a STA to find at least one AP. The channel switching time $T_{sw}(i)$ affects a STA when it starts to scan a new channel. Thus, these scanning parameters are considered as a group to affect the scanning delay. How to find a group that can satisfy the objective equation 4.10 subject to the constraints in Equation 4.11-4.13 can be considered as a combinatorial problem of non-linear objective and constraint functions. In this chapter, a GA algorithm is used to solve this problem because GA is well suited for solving non-linear objective and constraint functions expressed as both discrete and continuous variables [125].

4.3 Genetic Algorithm

In this section, the details about GA procedures and how to choose the GA operators in each procedure are introduced based on the scanning parameter optimization problems. Firstly, the chromosome representation and initializing the chromosome in the population is described. Then, the se-

Algorithm 7 Optimization of scanning parameters based on GA

```
1: $\alpha$: Population size;
2: $\beta$: Elitism rate, $\beta = 1/\alpha$;
3: $\gamma$: Crossover rate;
4: $\theta$: Mutation rate;
5: $\delta$: Number of generations;
6: Generate initial population $Pop$;
7: while termination conditions is not true do
8:     Calculate and evaluate fitness value of $Pop$ by Equation 4.14;
9:     Select the parents using the selection operators described in section 4.3.4;
10:    Generate new children by crossover with rate $\gamma$;
11:    Mutate the children with mutation rate $\theta$;
12:    Select $\alpha \times \beta$ of best solutions as elitism for next generation
13:    Update $Pop$ with new population;
14: end while
```

lection operator, crossover operator and mutation operation are discussed to select the most suitable operators that are suitable to the self-optimizing algorithm in real-time network conditions. Furthermore, how to evaluate the selected chromosomes for reproduction and how to terminate the GA process are presented. Finally, the optimized scanning parameters by GA also be further adjusted based on the dynamic changing network environments.

4.3.1 Chromosome Representation

The first aspect of a GA is how the solutions are encoded as chromosomes. A chromosome is a fixed-length string of genes represented for an individual. The genes can be binary or real-value numbers depending on the type of problems. The genes in a chromosome and the length of each chromosome depend on the parameters involved in the problems.

Conventionally, binary strings are used to represent the decision variables of the optimization problem in the genetic population, irrespective of

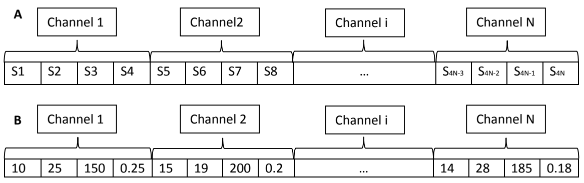


Figure 4.2: Chromosome representation: (A) chromosome encoding (B) an example of chromosome

the nature of the decision variables. A binary-coded GA has many difficulties in dealing with continuous search spaces in the scanning parameters optimization problem. The use of real value-coded (RVC) GA representation has many advantages over binary coding. The efficiency of the GA is increased as there is no need to convert the solution variables to the binary type, less memory is required, there is no loss in precision by discretization to binary or other values, and there is greater freedom to use different genetic operators [126].

Therefore, RVC GA is proposed in which the optimization variables are represented as the real-valued parameters. In the RVC representation, the scanning timers of all channels are considered genes. They are $t_{min}(i)$, $t_{max}(i)$, $T_{P_{req}}(i)$, $T_{sw}(i)$ and i is the channel number. Each gene represents a parameter in the problem. If the number of scanning channels is N, the size of each chromosome is an array of strings of length $N * 4$.

In Figure 4.2, a chromosome representation is shown. Each gene is indicated as below: S1 is denoted as $t_{min}(1)$, S2 is denoted as $t_{max}(1)$, S3 is denoted as $T_{P_{req}}(1)$, S4 is denoted as $T_{sw}(1)$, S5 is denoted as $t_{min}(2)$ and so on.

4.3.2 Population Initialization

The initial population described by Pop is a randomly generated set of chromosomes. The population size in our proposed GA is α (see line 1, 6 in Algorithm 7). Each chromosome is a set of scanning parameters. For example, if the number of channels is 11, then each channel i has four scanning parameters $t_{min}(i)$, $t_{max}(i)$, $T_{P_{req}}(i)$ and $T_{sw}(i)$. Thus, one chromosome has 44 genes. The initial values of scanning parameters $t_{min}(i)$, $t_{max}(i)$, $T_{P_{req}}(i)$ and $T_{sw}(i)$ are randomly generated by pseudorandom number generators.

4.3.3 Calculating Fitness Value

The fitness value is used to accurately evaluate the quality of the chromosomes in the population (see line 8 in Algorithm 7). The fitness value can be calculated using the fitness function given by Equation 4.14. It gives a fitness value to each chromosome. The chromosomes with high fitness values have more chances to be selected for reproduction. The probability that a chromosome will be selected for reproduction is based on the fitness value. The selected chromosomes will generate optimal solutions for scanning parameters until the termination condition is satisfied.

$$f_j = \frac{\sum_{i=1}^{N} SNR_h(i)}{\sum_{i=1}^{N} T_{sc}(i)}, \tag{4.14}$$

where f_j represents the fitness value of the j_{th} chromosome. The expectation of the fitness function is to use the lowest scanning time $\sum_{i=1}^{N} T_{sc}(i)$ to find more available APs with higher SNR values. The fitness value f_j is used for selection that will be introduced by section 4.3.4.

4.3.4 Selection

There are several selection methods to select the chromosome with the best fitness value. These include Tournament Selection, Proportional Roulette Wheel Selection, Rank-based Roulette Wheel Selection. The details of these selections are explained in the following subsections and the summary of the main three selection methods is show in Table 2.6.

For the scanning parameters optimization problem in WiFi networks, tournament selection is used to solve this issue. As tournament selection has a fast and efficient convergence rate, it will satisfy the real-time network requirements [67]. Also, the number of scanning parameters is not big and tournament selection is suitable for this small size problem [127]. After selection, the chromosome with the better fitness value will be selected as parents to produce new child chromosomes by the crossover and mutation operations. At the end of each generation, another selection method called elitism is employed. This method selects the best chromosome in the current generation to pass to the next generation. This action is important to ensure that the relevant features of the chromosomes, selected so far by the survival of the fittest principle, will not be discarded during the process of recombination (see lines 2, 9, 12 in Algorithm 7).

4.3.4.1 Tournament Selection

Tournament selection is considered probably the most popular selection methods in GA due to its efficiency and simple implementation [64]. As the scanning parameters need to be updated efficiently to reduce the scanning delay, the optimal value of scanning parameters needs to be obtained by GA as fast as possible to ensure the handover is timely and successful. Tournament selection also maintains steady pressure towards convergence [64]. Thus, tournament selection is chosen to evaluate the handover performance.

Figure 4.3 shows an example of tournament selection. K chromosomes are selected from the population at random and the best out of these are chosen to become a parent (see lines 2-3 in Algorithm 8). The same process is repeated for selecting the next parent. This process is repeated n times to produce the next generation of chromosomes (see lines 1-5 in Algorithm 8).

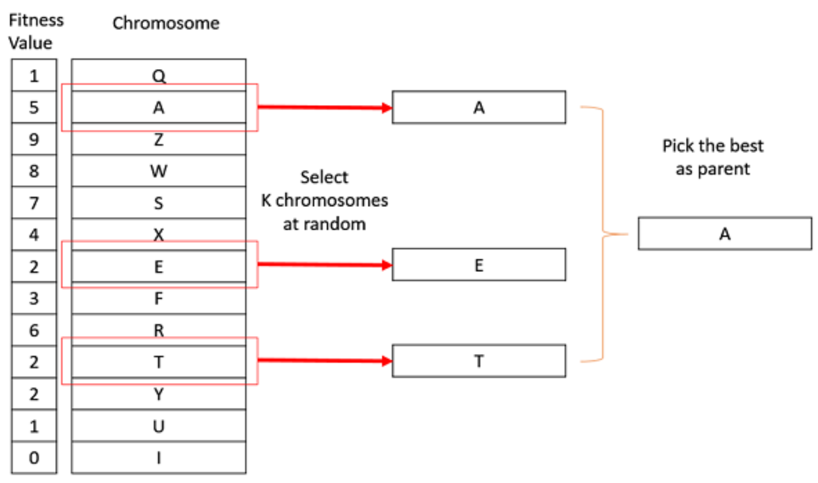


Figure 4.3: Tournament Selection

Algorithm 8 Tournament Selection

```
    **Input**: Population $P(t) = \{\alpha_1, ..., \alpha_n\}$ and tournament size: $k \in \{1, 2, ..., n\}$;
    **Output**: Population $P(t)' = \{\alpha_1', ..., \alpha_n'\}$ after selection
1: for $i \leftarrow 0$ to $n$ do
2:     randomly select $k$ chromosomes from $\{\alpha_1, ..., \alpha_n\}$;
3:     calculated the fitness value by Equation 4.14, the fitness values of the selected $k$ chromosomes are denoted as $\{f_1, ..., f_k\}$
4:     $\alpha_i' \leftarrow$ choose the chromosome with the maximum fitness value from $k$ randomly selected chromosomes as parent;
5: end for
6: return $\{\alpha_1', ..., \alpha_n'\}$
```

4.3.4.2 Roulette Wheel Selection

Roulette Wheel Selection was proposed by Holland [65] and assumes that the selection probability of an individual chromosome is directly related to fitness. In this selection, all the chromosomes in the population are placed on the roulette wheel according to their fitness value [71]. Each chromosome is assigned a segment of the roulette wheel whose size is proportional to the value of the fitness of the chromosome. Those chromosomes with the largest fitness (i.e. largest segment sizes) have a higher probability of being chosen. As Roulette Wheel Selection gives a chance to all chromosomes, it can help search for a global optimal group of scanning parameters. The accuracy of the selection method to make sure the scanning delay can be reduced and the best candidate AP can be found during search time. However, the time taken to compute the optimal scanning parameters is larger than the Tournament Selection.

Following are the steps for Roulette Wheel selection:

1. Calculate the fitness value of each individual denoted by f_i.

2. Compute the probability that an individual is chosen by dividing the individual's fitness by the sum of fitness values of the whole population.
$$p_i = \frac{f_i}{\sum_{i=1}^{n} f_j}; j = 1, 2, ..., n, \tag{4.15}$$
where n is the size of population.

3. Choosing one individual from the population with its probability p_i can be accomplished by defining its cumulative probability,
$$q_i = \sum_{j=1}^{i} p_j \tag{4.16}$$

4. Generate a uniform random number $r \in (0, 1]$ (see lines 3 in Algorithm 9).

5. If $r \leq q_1$, select the first chromosome, else select the ith indiviual such that $q_{i-1} \geq r \leq q_i$(see lines 3-9 in Algorithm 9).

6. Repeat steps 4-5 n times to create n candidates for new population.

Algorithm 9 Roulette Wheel Selection

```
    Input: Population $P(t) = \{\alpha_1, ..., \alpha_n\}$
    Output: Population $P(t)^{'} = \{\alpha^{'}_1, ..., \alpha^{'}_n\}$ after selection
 1: for $i \leftarrow 0$ to $n$ do
 2:     Calculate the cumulative probability by Equation 4.15, 4.16;
 3:     Generate a uniform random number $r \in (0, 1]$;
 4:     if $r \leq q_1$ then
 5:         $\alpha^{'}_i \leftarrow$ select the first chromosome from $\{\alpha_1, ..., \alpha_n\}$;
 6:     else
 7:         if $q_{i-1} \geq r \leq q_i$ then $\alpha^{'}_i \leftarrow$ select the $i$th chromosome;
 8:         end if
 9:     end if
10: end for
11: return $\{\alpha^{'}_1, ..., \alpha^{'}_n\}$
```

4.3.4.3 Ranked-based Selection

The roulette wheel selection has a problem when there are big differences between the fitness values in the initial population. If the best chromosome fitness is 90%, its circumference occupies 90% of the Roulette wheel, and then other chromosomes have too few chances to be selected [128]. This causes a premature convergence and a loss of diversity [129]. Thus, the optimal scanning parameters may not be found and the scanning delay will not be reduced.

The rank-based selection was introduced by Baker [74] to eliminate the disadvantages of the Roulette wheel selection. Rank-based selection is the selection strategy that the probability of a chromosome is selected based on its fitness rank in the entire population. Thus, the optimal value of scanning parameters could be explored without losing diversity. Compared with the roulette wheel selection, the rank-based selection is more robust towards the optimal value of scanning parameters [73]. In the rank-based selection scheme, the chromosomes in the population are ranked according to their fitness and then selection probabilities are computed according to their ranks rather than fitness values. Therefore, the rank-based selection can be computationally expensive because of the need to sort populations [129]. This long computation of soring fitness cannot ensure GA find the optimized scanning parameters fast enough to meet the dynamic network environment.

Algorithm 10 Linear Rank-based Selection

```
    Input: Population $P(t) = \{\alpha_1, ..., \alpha_n\}$
    Output: Population $P(t)' = \{\alpha_1', ..., \alpha_n'\}$ after selection
 1: for $i \leftarrow 0$ to $n$ do
 2:     Assign the fitness by the rank Equation 4.17;
 3:     Calculate the cumulative probability by Equation 4.15, 4.16;
 4:     Sort the population according the rank;
 5:     Generate a uniform random number $r \in (0, 1]$;
 6:     if $r \leq q_1$ then
 7:         $\alpha_i' \leftarrow$ select the first chromosome from $\{\alpha_1, ..., \alpha_n\}$;
 8:     else
 9:         if $q_{i-1} \geq r \leq q_i$ then $\alpha_i' \leftarrow$ select the $i$th chromosome;
10:         end if
11:     end if
12: end for
13: return $\{\alpha_1', ..., \alpha_n'\}$
```

In linear ranked-based selection, chromosomes are first sorted according to their fitness value and then the ranks are assigned to them (see lines 2-4 in Algorithm 10). The best chromosome is ranked at the last in the population and the worst one is ranked at the first in the population. The selection probability is then assigned linearly to the chromosomes according to their ranks (see lines 6-11 in Algorithm 10). The rank for a chromosome may be scaled linearly using the following formula,

$$Rank(Pos) = 2 - SP + (2.(SP - 1).\frac{(Pos - 1)}{n - 1}). \tag{4.17}$$

where Pos is the position of an individual in the population and SP is the selective pressure, which is the probability of the best individual selected by comparing to the average probability of selection of all individuals [130].

4.3.5 Crossover

The combination of two-parent chromosomes produces a new child chromosome and this is called a crossover. With the real value-coded representation, the evaluation procedure and reproduction operator remain the same as in binary-coded GA, but crossover and mutation operators for the RVC GA need to be redefined. For this scanning parameter optimization problem, real encoding is adopted because the continuous search spaces are needed when the scanning initiated periodically and high precision is required to make sure handover is timely and successful.

There are many types crossover for RVC GA including single-point crossover, N-points crossover and uniform crossover. In the N-points crossover, according to the number of crossover points, there are also two-points, three-points and so on.

As the representation shown in Figure 4.2 in the proposed GA method, the scanning parameters $t_{min}(i)$, $t_{max}(i)$, $T_{P_{req}}(i)$ and $T_{sw}(i)$ are considered as a group of parameters per channel. Therefore, the N-points crossover method is suitable to use for cutting the random position based on channel numbers. By the operation of crossover, the scanning parameters for each channel of two selected parents will be exchanged. The crossover rate is γ (see lines 3, 10 in Algorithm 7).

Let $P_1 = (p_1^1, ..., p_n^1)$ and $P_2 = (p_1^2, ..., p_n^2)$ denote two selected parents for crossover operation. The new offsprings are denoted as $C_1 = (c_1^1, ..., c_n^1)$ and $C_2 = (c_1^2, ..., c_n^2)$. Different crossover operators will be described in following sections.

4.3.5.1 Simple-point Crossover

Simple-point crossover is the earliest crossover operator used in the past [65,76]. This crossover uses the single-point fragmentation of the parents at the crossover point to create the child [131]. Simple point crossover first selects two parents used for a crossover (see line 3-4 in Algorithm 11). Then, randomly selects a crossover point k, $k \in \{1, ..., n-1\}$ where n is the number of channels. The crossover point is generated by pseudorandom number generators and the pseudorandom number should be an integer that is uniformly distributed between 1 and $n-1$. The crossover point is based on the channel numbers because the scanning parameters $t_{min}(i)$, $t_{max}(i)$, $T_{P_{req}}(i)$ and $T_{sw}(i)$ are considered as a group of parameters per channel in one chromosome as shown in Figure 4.2. Two new chromosomes are created by combining the parents at the crossover point (see line 5-8 in Algorithm 11), which are

$$\begin{aligned} C_1 &= (p_1^1, p_2^1, ..., p_i^1, p_{i+1}^2, ..., p_n^2), \\ C_2 &= (p_1^2, p_2^2, ..., p_i^2, p_{i+1}^1, ..., p_n^1). \end{aligned} \tag{4.18}$$

Algorithm 11 Simple-point crossover

```
   Input: Two parents chromosomes $P_1, P_2$
   Output: Two new chromosomes $C_1, C_2$
1: Simple-point-crossover($i, P_1, P_2$):
2: int $cp$;
3: Select two parents $P_1, P_2$ from a parent pool;
4: $cp \leftarrow$ randomly select a crossover point position by pseudorandom
   number generators following uniform distribution
5: for $i \leftarrow 0$ to $cp - 1$ do
       $C_1[i] \leftarrow P_1[i]$;
       $C_2[i] \leftarrow P_2[i]$;
6: end for
7: for $i \leftarrow cp$ to $n$ do
       $C_1[i] \leftarrow P_2[i]$;
       $C_2[i] \leftarrow P_1[i]$;
8: end for
9: return $C_1, C_2$
```

4.3.5.2 N-point Crossover

Algorithm 12 N-point crossover

```
    Input: Two parents chromosomes $P_1, P_2$
    Output: Two new chromosomes $C_1, C_2$
 1: N-point-crossover($i, P_1, P_2$):
 2: Select two parents $P_1, P_2$ from a parent pool;
 3: local array $Points$;
 4: $Points \leftarrow$ randomly select $k$ different crossover points by pseudoran-
    dom number generators following uniform distribution
 5: for $i \leftarrow 0$ to $n-1$ do
 6:     if $j < k, i = Points[j]$ then
 7:         $j \leftarrow j+1$;
 8:     end if
 9:     if $j$ is an even number then
          $C_1[i] \leftarrow P_1[i]$;
          $C_2[i] \leftarrow P_2[i]$;
10:     else
          $C_2[i] \leftarrow P_1[i]$;
          $C_1[i] \leftarrow P_2[i]$;
11:     end if
12: end for
13: return $C_1, C_2$
```

N-point crossover uses the point to combine the parents the same as per simple point crossover. N-Point Crossover first selects parents used for crossover (see line 2 in Algorithm 12). Then randomly selects n crossover points [132]. The N crossover points are selected based on the number of channels because the scanning parameters $t_{min}(i)$, $t_{max}(i)$, $T_{P_{req}}(i)$ and $T_{sw}(i)$ are a group of parameters per channel in one chromosome. Let k denote the number of crossover points. The k different crossover points are generated by pseudorandom number generators and the pseudorandom number should be an integer that is uniformly distributed between 1 and $n-1$ where n is the number of channels (see line 3-4 in Algorithm 12). Two parents combine at the crossover points and generate new children. Compared with single-point crossover, N-point crossover has a wide search space that can find global optimal scanning parameters for the self-optimizing problem. This is because the single-point crossover only selects one cross point to combine the parents to produce new child chromosomes and makes it harder to move out of the local optimum.

Taking $N = 2$, two crossover points are selected randomly at position i and j ($i, j \in \{1, 2, ..., n-1\}$ with $i \leq j$), with i and j being different numbers of channels as described in section 4.3.5.1. Two new chromsomes are created (see line 5 - 12 in Algorithm 12) as follows:

$$\begin{aligned} C_1 &= (p_1^1, p_2^1, ..., p_i^2, p_{i+1}^2, ..., p_j^2, p_{j+1}^1, ..., p_n^1), \\ C_2 &= (p_1^2, p_2^2, ..., p_i^1, p_{i+1}^1, ..., p_j^1, p_{j+1}^2, ..., p_n^2). \end{aligned} \tag{4.19}$$

4.3.5.3 Uniform Crossover

Algorithm 13 Uniform crossover

```
   **Input**: Two parents chromosomes $P_1, P_2$
   **Output**: Two new chromosomes $C_1, C_2$
1: Uniform-crossover($i, P_1, P_2$):
2: Select two parents $P_1, P_2$ from a parent pool;
3: Define a value for probability $p_c$.
4: for $i \leftarrow 0$ to $n$ do
5:     Caculate each chromosome vector $c_i^k$ by Equation 4.20;
6: end for
7: return $C_1, C_2$
```

In contrast to previous crossover operators, the uniform crossover operator does not divide the parent chromosome into segments for recombination. Rather, it provides the uniformity in combining both parents [131]. The uniform crossover first selects two parents used for crossover (see line 2 in Algorithm 13). Two new chromosomes can be described as $C_k = (c_1^k, ..., c_i^k, ..., c_n^k), k = 1, 2$. Each chromosome vector c_i^k can be calculated by Equation 4.20,

$$c_i^k = \begin{cases} p_i^2 & \text{if } u = 0 \\ p_i^1 & \text{if } u = 1 \end{cases}, \tag{4.20}$$

where u is a random number that can have a value of zero with probability p_c, and a value of one with probability $1 - p_c$; With probability p_c the value of the first parents gene p_i^1 is assigned to the second child and the value of the second parents gene p_i^2 is assigned to the first child. The new children can be created by line 4-7 in Algorithm 13. An example of this operator has been used successfully by Gonalves et al. [133] with probability $p_c = 0.7$.

As the uniform operator can distribute solutions widely across the search space, it avoids convergence to a local optimum. In the self-optimizing problem, the uniform operator can make sure to find global values of the scanning parameters to reduce scanning delay. However, the fixed probability of p_c cannot satisfy the dynamic network requirements to find the optimal scanning parameters fast enough. The reason is that the probability of p_c controls the number of genes swapping between the two parents. With the number of generation increases, the genes of two parents are more and more similar. Thus, swapping probability should be reduced. The fixed probability p_c may increase the time complexity.

4.3.6 Mutation

Mutation operator in GA is used primarily as a mechanism for maintaining diversity in the population [80]. In contrast to a crossover operator, a mutation operator operates on only one population member at a time and modifies it independent to the rest of the population members. Although mutation operator alone may not constitute an efficient search, along with a suitable crossover operator, mutation operator plays an important role in making the overall search efficient [80].

There are several types of mutation such as random mutation, boundary mutation, and non-uniform mutation. As boundary mutation only selects the upper or lower value of the user-defined boundaries to replace a gene, the diversity of scanning parameters cannot be satisfied in the self-optimizing problem. Thus, boundary mutation cannot ensure to find the global optimums of scanning parameters to reduce scanning delay. The parameters used in the non-uniform mutation have to be tuned with the generation increasing. The tuning time can increase the scanning delay that is not suitable to solve the self-optimizing problem which aims to reduce the scanning delay.

Compared with boundary mutation and Non-uniform mutation, random mutation replaces the gene with a uniformly distributed random value. As a gene is a scanning parameter, the randomly selected value increases the diversity of the scanning parameters. The diversity can make sure the optimal value of scanning parameters can be found in the global search space. Thus, random mutation is considered to be one of the most suitable mutation operators for RVC GA [134] and used to solve the self-optimization problem.

In the Random mutation, a random gene is selected and its value is changed within the entire parameter range. The mutation rate is θ (see lines 4, 11 in Algorithm 7). The example is given in Figure 4.4.

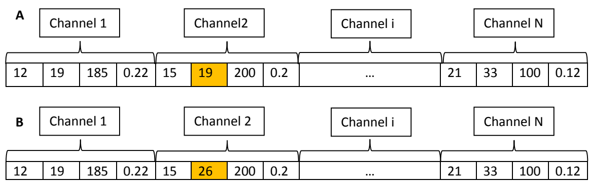


Figure 4.4: Mutation: (A) before mutation (B) after mutation

Let $C = (c_1, ...c_i, ...c_n)$ denote a chromosome and $c_i \in [a_i, b_i]$ denote a gene to be mutated. Next, the gene, $c_i^{'}$, resulting from the application of different mutation operators is shown in following subsections.

4.3.6.1 Random Mutation

Michalewicz [81] proposed a random (uniform) mutation based on floating point representation. The random mutation operator replaces the value of the chosen gene with a random value (uniform probability distribution) selected between the user-specified upper and lower bounds for that gene [82] (see line 2-5 in Algorithm 14). This operator ensures that the GA can search the solution space freely [83]. Thus, the global optimums of scanning parameters can be found to be used in the scanning phase. The handover delay can be reduced with the optimal scanning parameters.

Algorithm 14 Random mutation

```
    Input: Chromosomes $C = (c_1, ...c_i, ...c_n)$ before mutation
    Output:Chromosomes $C^{'} = (c_1, ...c_i^{'}, ...c_n)$ after mutation
1: Random-mutation($C$):
2: Select a random integer number $i$ from $[1, n]$;
3: $a_i$ is the lower bound of the $i$-th scanning parameter;
4: $b_i$ is the upper bound of the $i$-th scanning parameter;
5: Set $c_i \in [a_i, b_i]$
6: return $C^{'}$
```

4.3.6.2 Boundary Mutation

Boundary mutation is variation of the uniform mutation with $c_i^{'}$ being either the lower bound a_i or upper bound b_i with equal probability (see line 2-11 in Algorithm 15). This operator cannot meet the diversity of scanning parameters in GA because the genes are mutated only by boundaries. With the generation increases, the chromosomes become less different. The gene mutated only with boundary values means the chromosomes have less opportunity for mutation. The chromosomes may be the same in the later generations. Thus, it is a waste of time to do mutation. The time consumption cannot meet the dynamic network requirements to find the best scanning parameters as fast as possible. Thus, the best solutions

Algorithm 15 Boundary mutation

```
    Input: Chromosomes $C = (c_1, ...c_i, ...c_n)$ before mutation
    Output:Chromosomes $C^{'} = (c_1, ...c_i^{'}, ...c_n)$ after mutation
 1: Boundary-mutation($C$):
 2: Select a random integer number $i$ from $[1, n]$;
 3: $r$ is a random selected value between 0 and 1.
 4: Select a random real value $r$ from (0,1)
 5: $a_i$ is the lower bound of the $i$-th scanning parameter;
 6: $b_i$ is the upper bound of the $i$-th scanning parameter;
 7: if $r > 0.5$ then
 8: $c_i^{'} \leftarrow b_i$;
 9: else
10: $c_i^{'} \leftarrow a_i$;
11: end if
12: return $C^{'}$
```

of scanning parameters may not be guaranteed using boundary mutation. If the selected scanning parameters are not the best solutions, the scanning delay cannot be reduced and handover delay will be high.

4.3.6.3 Non-uniform Mutation

Algorithm 16 Non-uniform mutation

```
    Input: Chromosomes $C = (c_1, ...c_i, ...c_n)$ before mutation
    Output:Chromosomes $C^{'} = (c_1, ...c_i^{'}, ...c_n)$ after mutation
 1: Non-uniform-mutation($C$):
 2: Select a random integer number i from $[1, n]$;
 3: Select a random real value $r$ from (0,1)
 4: $a_i$ is the lower bound of the $i$-th scanning parameter;
 5: $b_i$ is the upper bound of the $i$-th scanning parameter;
 6: Set $c_i^{'}$ by Equation 4.21;
 7: return $C^{'}$
```

Non-uniform mutation is one of the commonly used mutation operators in real coded GAs [81] . If this operator is applied in a generation t, and g_{max} is the maximum number of generations then

$$c_i^{'} = \begin{cases} c_i + \Delta(t, b_i - c_i) & \text{if } \tau = 0 \\ c_i - \Delta(t, c_i - a_i) & \text{if } \tau = 1 \end{cases} \tag{4.21}$$
$$\Delta(t, y) = y(1 - r^{(1-\frac{t}{g_{max}})^{\sigma}}),$$

where τ is a random value which may be zero or one, and r is a uniformly distributed random number from the interval $[0, 1]$, σ is a parameter chosen by the user, which determines the degree of dependency on the number of iterations. The function $\Delta(t, y)$ gives a value in the range $[0, y]$ such that the probability of returning a number close to zero increases as t increases. In the initial generations of non-uniform mutation tends to search the space uniformly and in the later generations, it tends to search the space locally [135]. This can ensure the GA to find the good solutions of scanning parameters in the global searching space and refine the global solutions accurately in local space. Although the accuracy of finding the optimal scanning parameters is high using non-uniform mutation, the calculation time of the operator is long. This does not satisfy the objective to reduce the scanning delay.

4.3.7 Termination

The GA process is terminated in two ways. The first way is when the best AP is discovered before the maximum generation is reached. When the SNR_h defined in Equation 4.9 of a certain AP is higher than a pre-defined maximum threshold SNR_{up}, the GA process and active scan are terminated. The current chromosome will be selected as the best chromosome. The second way is that the GA will continue to run until the last generation.

$$N_{gen} = \begin{cases} j, & \text{if } SNR_h \geq SNR_{up} \\ N_{max}, & \text{otherwise} \end{cases} \tag{4.22}$$

where N_{gen} is the generation needed to run GA procedures, j is the j_{th} generation when $SNR_h \geq SNR_{up}$ is satisfied before the best chromosome is found, N_{max} is the maximum generation size.

4.3.8 Sorting Selected Channels

To provide seamless handover for real-time applications, reducing scan time for each channel is not enough. The total scanning time is affected by both the probe waiting time and the channel switching time when $T_{P_{req}}(i)$ is fixed, described in Equation 4.2. Thus, after the GA process, the channel will be sorted according to the value of $SNR_h(i)$.

There are two thresholds, a pre-defined maximum threshold SNR_{up} and a pre-defined minimum threshold SNR_{low}. The channel will be ranked and selected between two thresholds SNR_{low} and SNR_{up}. For the active scan, only the selected channel will be scanned by a STA. The reduction of the number of channels will shorten the scan time. At the same time, the sorted channel list will help to find the best candidate AP faster than the original scan sequence.

Let $C(ch)$ denote the set of scanning channels whereby each channel is indexed by i such that $1 \leq i \leq n$ and $i \in \mathbb{Z}^+$. Using a set notation we express the channel set as:

$$C_i(ch) = \{ch_1, ..ch_k, ...ch_n\}, ch_k \in N, \tag{4.23}$$

where ch_k denotes the channel with index k. The update algorithm finds a natural sorted set [118] (denoted by $C_i^o(ch)$) based on the three radio parameters (SNR, packet delay and load). The IEEE 802.11k standard provides mechanisms to retrieve these radio parameters from measurement reports such as SNR (signal to noise ratio), load and neighbor report. After retrieving these values from measurement reports, the list of channels is sorted from best to worst potential APs for handover. This information is recorded in a table stored in the controller.

4.4 Adaptive Timers

After setting the scanning timers by GA, these timers are dynamically adapted using Algorithm 17.

Algorithm 17 describes the dynamic adaptation of probe timers based on GA. With the information of GA readily known, the value of the timers *MinChannelTime* and *MaxChannelTime* for the channels on the GA list is optimized to reduce the wasted channel probing time (lines 12 - 15, 17 - 20 in Algorithm 17), because we already have the sorted channel information.

Algorithm 17 Dynamic adaptation of *MinChannelTime* and *MaxChannelTime*

```
1: $Channel(i)$: channel $i$;
2: $t_{min(i)}$: $MinChannelTime$ of channel $i$;
3: $t_{max(i)}$: $MaxChannelTime$ of channel $i$;
4: $t_{P_{req}(i)}$: Probe request interval $T_{P_{req}}(i)$;
5: $t_{sw(i)}$: Channel switching time $T_{sw}(i)$;
6: $t_{min(i)_{new}}$: Optimized $MinChannelTime$;
7: $t_{max(i)_{new}}$: Optimized $MaxChannelTime$;
8: $t_{P_{req}(i)_{ga}}$: GA Optimized probe request interval;
9: $t_{sw(i)_{ga}}$: GA Optimized channel switching time;
10: for GA selected sorting $Channel(i)$ do
11:     if AP is found then
12:         $t_{min(i)} \leftarrow \min\{0, t_{min(i)_{new}} - \alpha\}$;
13:         $t_{max(i)} \leftarrow \min\{0, t_{max(i)_{new}} - \beta\}$;
14:         $t_{P_{req}(i)} = t_{P_{req}(i)_{ga}}$;
15:         $t_{sw(i)} = t_{sw(i)_{ga}}$;
16:     else
17:         $t_{min(i)} \leftarrow \min\{t_{min(i)_{new}} + \alpha, \gamma\}$;
18:         $t_{max(i)} \leftarrow \min\{t_{max(i)_{new}} + \beta, \gamma\}$;
19:         $t_{P_{req}(i)} = t_{P_{req}(i)_{ga}}$;
20:         $t_{sw(i)} = t_{sw(i)_{ga}}$;
21:     end if
22: end for
```

When new APs are discovered in a scan, the timers of $t_{min(i)}$ and $t_{max(i)}$ are reduced by α and β respectively to avoid excessive waiting time (see

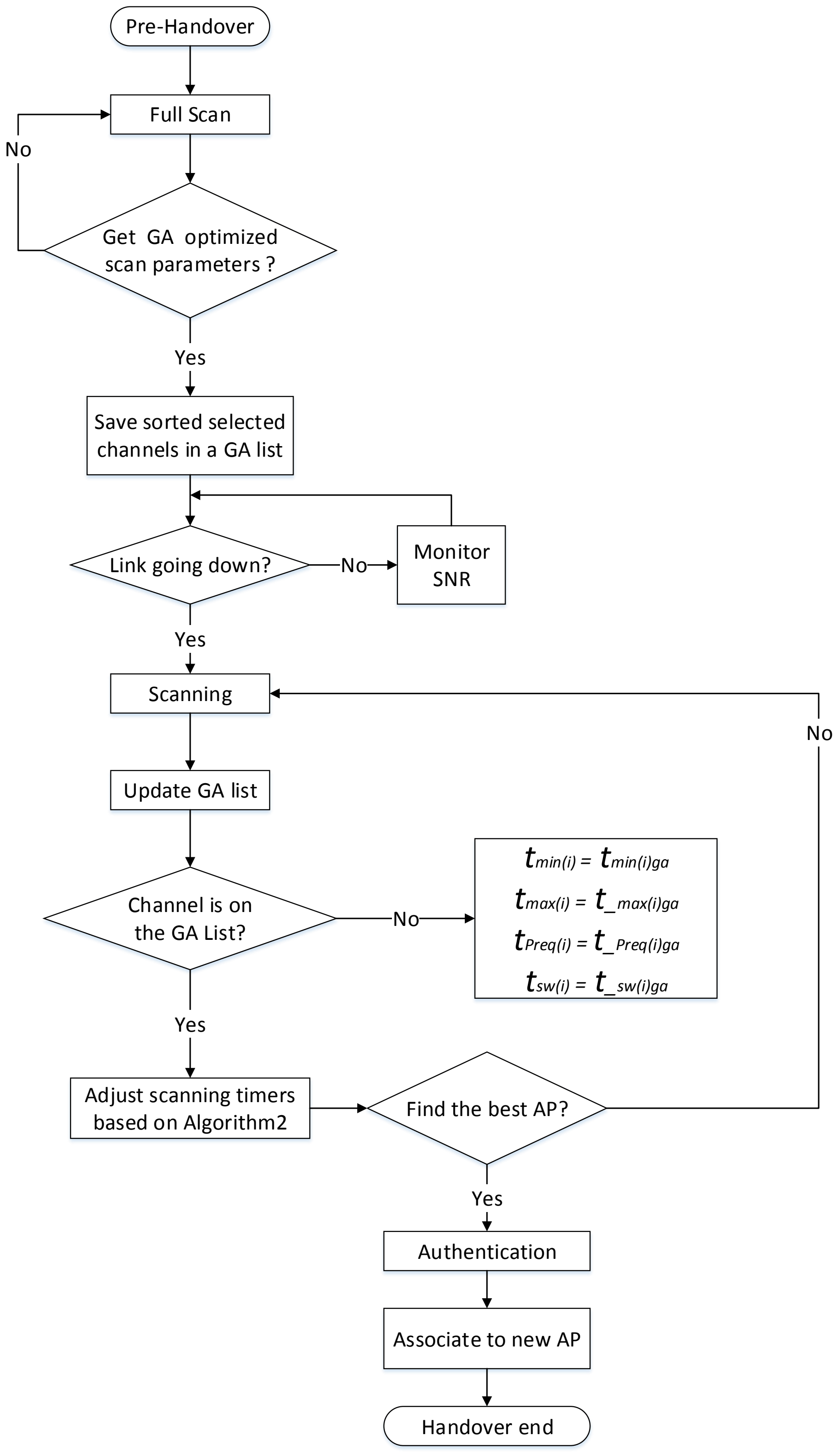

Pre-Handover
Full Scan
No
Get GA optimized scan parameters ?
Yes
Save sorted selected channels in a GA list
Link going down?
No
Monitor SNR
Yes
Scanning
No
Update GA list
Channel is on the GA List?
No
$t_{min(i)} = t_{min(i)ga}$
$t_{max(i)} = t_{\_max(i)ga}$
$t_{Preq(i)} = t_{\_Preq(i)ga}$
$t_{sw(i)} = t_{\_sw(i)ga}$
Yes
Adjust scanning timers based on Algorithm2
Find the best AP?
Yes
Authentication
Associate to new AP
Handover end

Figure 4.5: Flowchart of self-optimizing handover scheme

lines 12 - 15 in Algorithm 17). For existing APs in the list that have not replied to a probe request, the timers are increased to avoid missing out potential APs (see lines 17 - 20 in Algorithm 17). The values of α and β are the step increments for the timers. In this thesis these values are constant, γ is the maximum allowed scanning time specified by the 802.11 standard (12ms). After the scanning phase, we obtain the best AP for reassociation. The complete proposed handover process is illustrated in Figure 4.5.

4.5 Performance Evaluation

In this section, the performance of the proposed self-optimizing approach for scanning parameters is evaluated compared with the 802.11 standard using the fixed timers suggested in [26] and the previous neighbor list mechanism (NLM) handover scheme [136]. The reason why the proposed self-optimizing approach is compared with NLM is to evaluate whether the performance using dynamic scanning parameters is better than using fixed values in NLM.

4.5.1 Simulation Setup

The proposed algorithms are implemented in NS-3 with all AP operating in the 2.412 GHz band using the IEEE 802.11k protocol. The values of simulation parameters are listed in Table 4.1. The measurement data of the handover delay, throughput and packet loss rate are collected over 35 runs. Each run uses a different seed to generate a random data rate and packet size. The random traffic generation aims to satisfy different application requirements including voice call, video call, video streaming, online gaming in the different density network environment and network topologies. The averages shown are reported with a confidence interval of 95.00% under the assumption that the averages are normally distributed. This statistical methodology validates the simulation results. The simulation results are also validated by comparing proposed algorithms with conventional 802.11 standard and NLM using the same simulation parameters.

Table 4.1: Simulation Parameters

Parameter	**Value**
Simulation time (t)	60s, 100s
Speed	1m/s - 10m/s
Number of AP	6 - 30
Distance between two APs	30 m
Number of Mobile Nodes	1-35
ActiveProbing	true
WiFi Standard	802.11g, 802.11k
Packet size	10-10000 byte
Maximum data rate	54 Mbps
GA population size	50
GA Maximum number of generations	1000
GA Crossover rate	0.80
GA Mutation rate	0.15
GA Elitism rate	0.02

In the simulation, five different network scenarios are investigated. The first two scenarios involve six APs and five STAs. The APs are placed on a line topology and each AP is separated from its neighbors by 30 meters. These two scenarios are investigated by changing STAs speed and moving directions. The third scenario investigates the density of the network by changing the number of APs. The fourth scenario investigates the impact on handover performance by varying the number of STAs. Finally, three different network topologies of AP placement are compared to show the handover performance comparison. These three topologies are line topology, grid topology and random topology. All five scenarios reflect a typical indoor office or campus WLAN.

In order to evaluate the Self-Optimizing approach by different GA operators, three GA algorithms using different selection operators, crossover operators and mutation operators are used to compare the results with 802.11 standard and NLM handover scheme. These GA algorithms are GA-TNR using Tournament selection, N-point crossover and Random mutation, GA-RSR using Roulette Wheel Selection, Single-point crossover and Random mutation, GA-RNB Ranked-based Selection, N-point crossover and Boundary mutation.

4.5.2 Analysis of Results

4.5.2.1 Changing Speed of STAs

In the first scenario (Scenario I), the handover performance is evaluated by varying the speed of STAs. The speed of STAs is increased from 1m/s to 6m/s. At the simulation start time $t = 0$, the STAs are on the same location and are connected to AP0. All the STAs start to move at simulation time $t = 0$. The simulation time is 60s for each run.

- **Throughput**

  The average throughput has been compared when varying the speed of STAs to better understand the handover performance using different self-optimizing algorithms. The average throughput decreases with the moving speed increasing in all mechanisms. This is because the increased speed of STAs causes more handover frequencies. However, in the case of the proposed self-optimizing handover scheme, throughput is less affected by the increase of speed compared with standard 802.11.

  As can be seen in Figure 4.6, the average throughput of IEEE 802.11 drops 9.06% when the speed change from 2m/s to 6m/s. The NLM

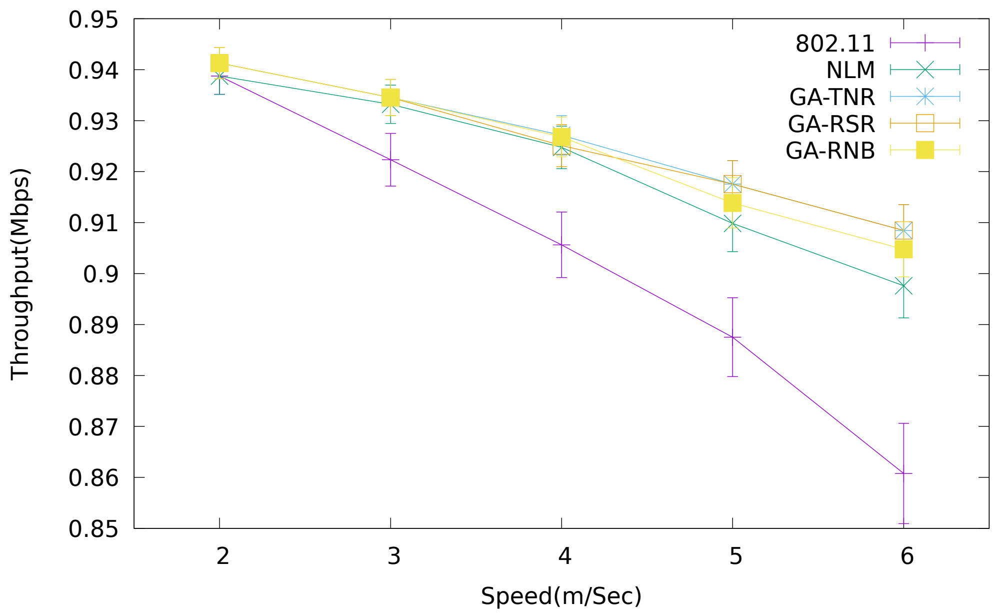


Figure 4.6: Throughput vs. Speed when the speed of STAs is changed (Scenario I)

method has improved the average throughput by around 1.97% compared with 802.11. However, with the proposed self-optimizing approach, the average throughput has been improved by 2.52% using GA-TNR, 2.48% using GA-RSR and 2.35% using GA-RNB, respectively. The average throughput performance improved is because the self-optimizing scanning parameters are dynamically optimized based on the network environment changing. Although the speed of STAs has been increased, the scanning parameters can be dynamically adjusted by the changes in speed. Thus, even though increasing the speed can cause more handovers and data disruption, the scanning delay has been reduced to adapt to the network changes.

Although the NLM method has already enhanced the overall average throughput, the GA based self-optimization approaches still show better performance than NLM when the speed of STAs is increased to five and six. This is because the self-optimizing approach not only reduces the number of scanning channels but optimizes the scanning parameters based on the dynamic network environment. The scanning parameters are adjusted when the speed of STAs increases. These adaptive parameters can satisfy the dynamic changing environment.

The GA-TNR shows the best throughput performance in this speed changing scenario. This is because tournament selection has a fast and efficient converge rate to satisfy the speed changing requirements. Moreover, N-points crossover is more suitable for this scanning parameter optimization because these parameters are considered as a group of parameters per channel. Random mutation is also one of the most suitable mutation operations for RVC GA and can search the solution space freely [83]. The throughput improvement of GA-RSR is 0.13%, which is worse than GA-TNR. This is because the roulette wheel selection in GA-TNR has a longer computing time shown in Table 2.6 than tournament selection. GA-RNB shows worse improvement than GA-TNR and GA-RSR because the boundary mutation cannot maintain the diversity of scanning parameters in GA. The boundary mutation just randomly selects the upper or lower value to replace the current gene of the chromosome. As the gene is the scanning parameter, the scanning parameter is not limited to the upper or lower value. Thus, the boundary mutation cannot make the overall search efficiently and the best solutions of scanning parameters may not be found.

- **Packet loss rate**

  Figure 4.7 illustrates the impact of handover on the data reception in different methods: 802.11, NLM GA-TNR, GA-RSR and GA-RNB. The average packet loss rate for standard 802.11, NLM, GA-TNR, GA-RSR and GA-RNB are 6.03%, 4.16%, 3.64%, 3.65%, 3.81%, respectively. The NLM method improved 31.01% compared with stan-

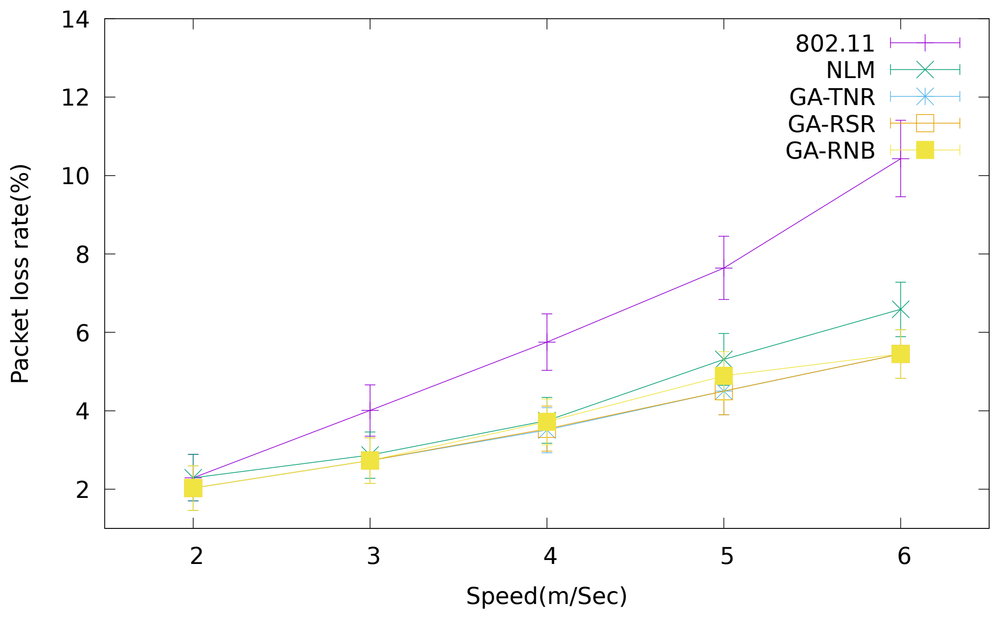


Figure 4.7: Packet loss rate vs. Speed when the speed of STAs is changed (Scenario I)

dard 802.11. The proposed self-optimizing algorithms using GA-TNR, GA-RSR and GA-RNB improved 39.64%, 39.47%, 36.82%, respectively. The GA-TNR achieves the best performance among these five approaches is corresponding to the results of throughput. This is because tournament selection can converge very fast and efficiently to satisfy the speed changing environment. Furthermore, N-points crossover is quite suitable for the scanning parameters cutting as a group per channel. Random mutation can maintain the diversity of the scanning parameters and also make the overall search efficient. As GA-RNB uses the boundary mutation that cannot make the diversity of scanning parameters in GA, the improvement of the packet loss rate is worse than GA-TNR and GA-RSR. Compared with NLM, the performance of packet loss rate are further improved by 8.63% with GA-TNR, 8.46% with GA-RSR, 5.81% with GA-RNB, respectively. This reducing is because both the number of channels scanned and the scanning parameters have been dynamically adjusted when the network environment changes as discussed in section 4.3.

- **Handover delay**

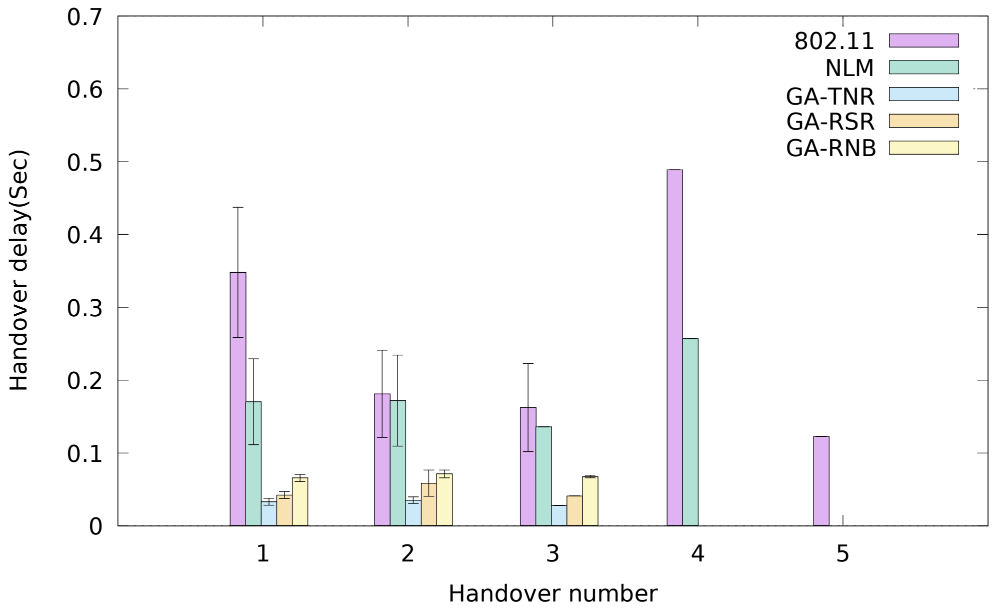


Figure 4.8: Handover delay vs. Handover number when the speed of STAs is changed (Scenario I)

Figure 4.8 shows the average handover delay impacted by varying STAs speed. The number of handovers certainly increases with increased moving speeds. However, compared with standard 802.11 and NLM, the average number of handovers of the proposed self-optimizing approach is reduced to three. The standard 802.11 and NLM method shows five and four handovers for 802.11 and NLM respectively on average. The reduced number of handovers of the self-optimizing approach means the STAs can find a better AP to associate with than 802.11 and NLM. Although the speed of STAs changes, the STAs can find the best candidate AP within a short scanning time using the optimized scanning parameters. Also, the channels are sorted from best to worst potential APs for handover as discussed in section 4.3.8. The number of handovers is reduced because the STAs are associated with a better AP searched by self-optimizing algorithm than the AP found based on the 802.11 standard.

The standard layer 2 handover scheme exhibits a lower performance, with an average handover delay of 0.33s/ho (handover numbers). The average handover delay of the proposed self-optimizing method is 0.04s/ho, 0.05s/ho and 0.07s/ho by using GA-TNR, GA-RSR and GA-RNB, respectively. Although the NLM improves to 0.19s/ho, the self-optimizing still achieves a better handover performance than NLM because the number of scanned channels and scanning timers is both optimized, while NLM only reduces the set of scanning APs. Compared with GA-RSR and GA-RNB, GA-TNR uses tournament selection, N-point crossover and random mutation operators that are most suitable to solve the scanning parameters optimization problem. This is because tournament selection is suitable for this small data size optimization problem. Also N-point crossover can be used to cut the scanning parameters as a group of parameters per channel. Furthermore, random mutation can maintain the diversity of scanning parameters to enhance the searching efficiency.

4.5.2.2 STAs Moving in Random Directions

In the second scenario (Scenario II), the STAs move at a constant speed but in random directions. At the simulation start time $t = 0$, the STAs are at different locations and are connected to different APs. All the STAs start to move at simulation time $t = 0$. The simulation time is 60s for each run.

- **Throughput**

  In Figure 4.9, the average throughput of standard 802.11 has been improved by around 2.64% using the NLM method by reducing the number of channels scanned.

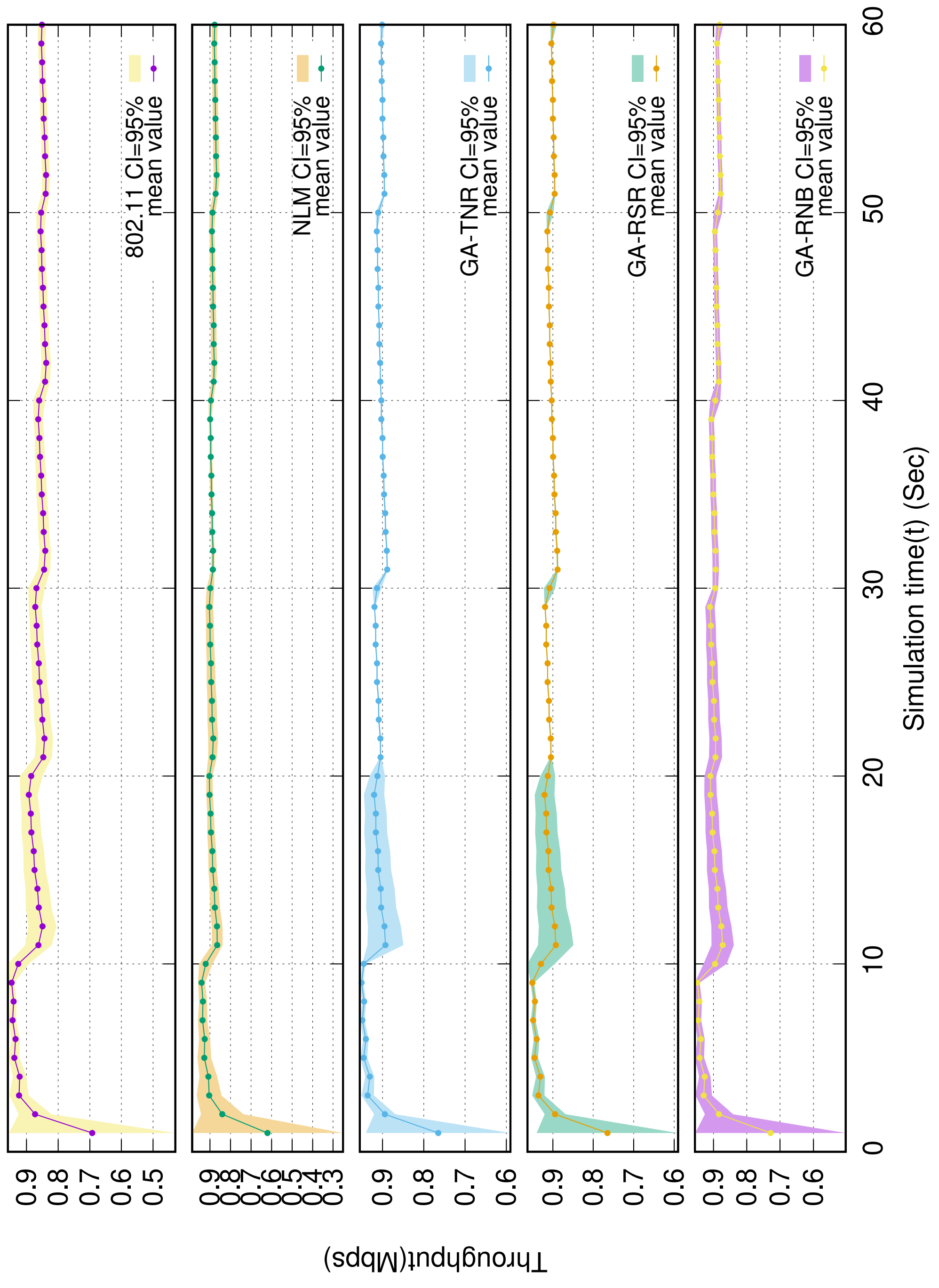


Figure 4.9: Throughput vs. Simulation time when the STAs move randomly (Scenario II)

With the proposed self-optimizing approaches, the average throughput has been improved by 5.10% with GA-TNR, 5.05% with GA-RSR and 4.75% with GA-RNB, respectively. All the GA optimization algorithms perform better than NLM. Compared with NLM, the average throughput is further enhanced. This is because the self-optimizing approaches not only optimize the number of channels scanned by a STA but also automatically adjust the scanning parameters.

With the self-optimizing approach, the STAs can detect the best candidate AP faster than 802.11 standard and NLM. GA-TNR achieves better throughput performance than GA-RSR and GA-RNB because the tournament selection, N-point crossover and random mutation operators used in GA-TNR can converge quickly to find the optimal values of scanning parameters as discussed in section 4.3. The roulette wheel selection used in GA-RSR and ranked-based selection used in GA-RNB has larger time complexity as shown in Table 2.6 than tournament selection used in GA-TNR. Also, the single-point crossover in GA-RSR has less searching space than N-point crossover and boundary mutation in GA-RNB cannot maintain the diversity in the population of scanning parameters as discussed in section 4.3.5.2 and 4.3.6.2.

- **Packet loss rate**

  Figure 4.10 illustrates the impact of handover on the data reception in the five methods: 802.11, NLM, GA-TNR, GA-RSR and GA-RNB. These five packet loss rates are 10.10%, 7.70%, 5.52%, 5.57% and 6.73% respectively on average. The proposed self-optimizing approaches achieve average 4.58% , 4.53% and 3.37% lower packet loss rate compared with the standard 802.11 mechanism by using GA-TNR, GA-RSR and GA-RNB respectively. With the reduced number of channels in NLM, the packet loss rates have already been reduced. However, the packet loss rate of the proposed self-optimizing ap-

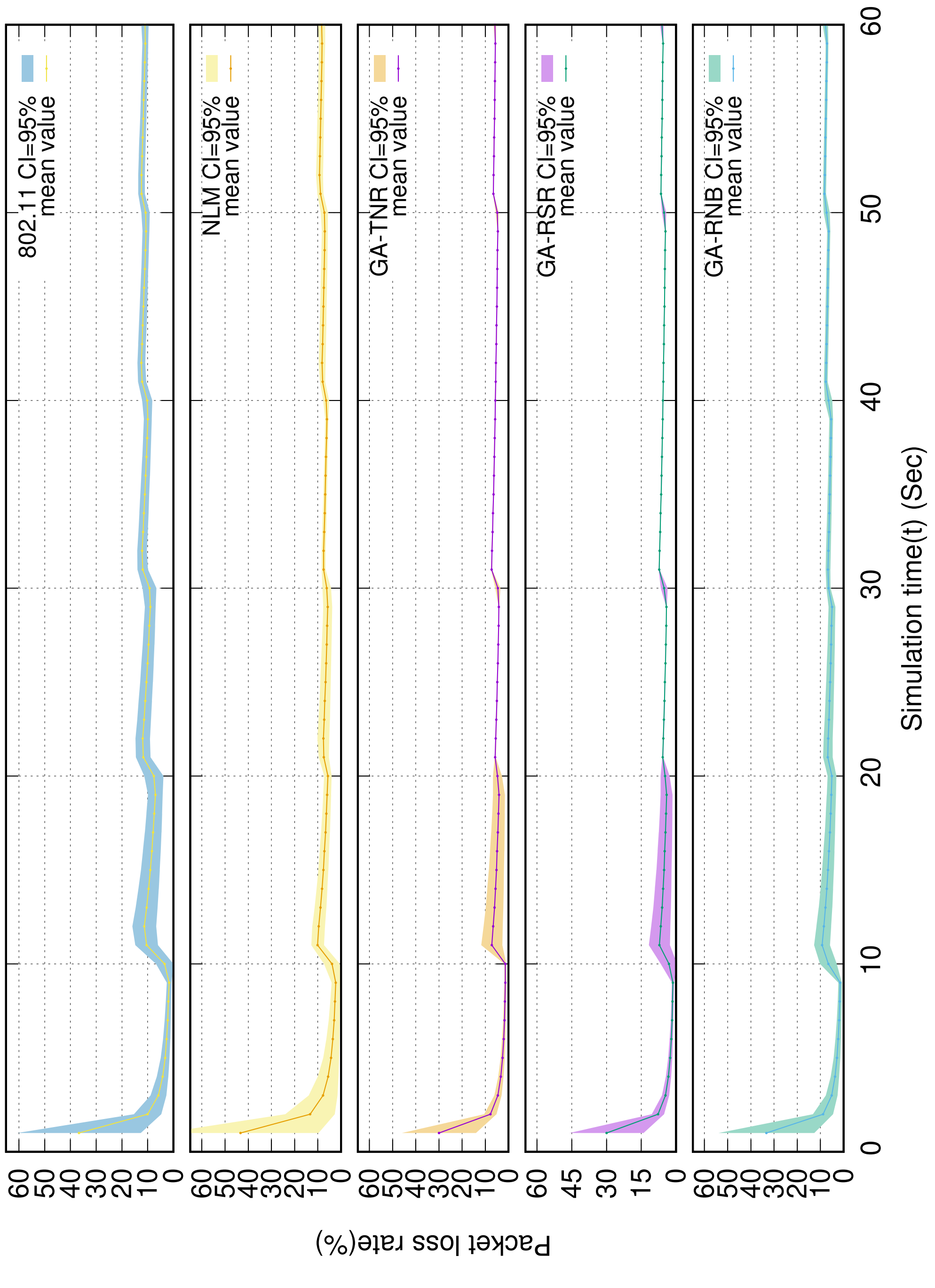


Figure 4.10: Packet loss rate vs. Simulation time when the STAs move randomly (Scenario II)

proaches drops by 28.31% with GA-TNR , 27.66% with GA-RSR and 12.60% with GA-RNB compared to NLM because the self-optimizing timers further optimized the scanning parameters which in turn provides more responsive handovers. The GA-TNR achieves the best performance among these five approaches. This is because tournament selection used in GA-TNR is more suitable for this small data size optimization problem. Moreover, N-points crossover is suitable for the scanning parameters cutting as a group per channel and random mutation can make the population of scanning parameters more diverse. As GA-RNB uses the boundary mutation using the upper and lower boundary values to mutate the genes, it cannot make the diversity of scanning parameters in GA discussed in section 4.3.6.2. Thus, the improvement of the packet loss rate is worse than GA-TNR and GA-RSR.

- **Handover delay**

  Figure 4.8 shows the total handover delay during the simulation time. Compared with the 802.11 standard, the handover delay is reduced by 15.56% by using NLM. The handover of self-optimizing approaches is reduced by 29.46% with GA-TNR, 29.19% with GA-RSR and 27.03% with GA-RNB compared with NLM, respectively. With the reduction of total handover delay, the data disruption time is reduced to provide seamless handover performance. When the STAs move randomly, the GA-TNR method can find the best candidate AP faster than GA-RSR and GA-RNB because tournament selection can converge faster to the optimal value. Also N-point crossover is more suitable to this scanning parameter optimization problem because the scanning parameters are considered as a group of parameters per channel. Furthermore, the random mutation that follows the uniform distribution probability can select a value for a gene between the upper and lower boundary. However, the boundary mutation used in GA-RNB can only choose the upper or lower boundary value to replace the value of genes. GA-RNB cannot keep the diversity of the population and affect the efficiency of the solutions search space. Thus, the handover delay reduced by GA-RNB is less than GA-TNR and GA-RSR.

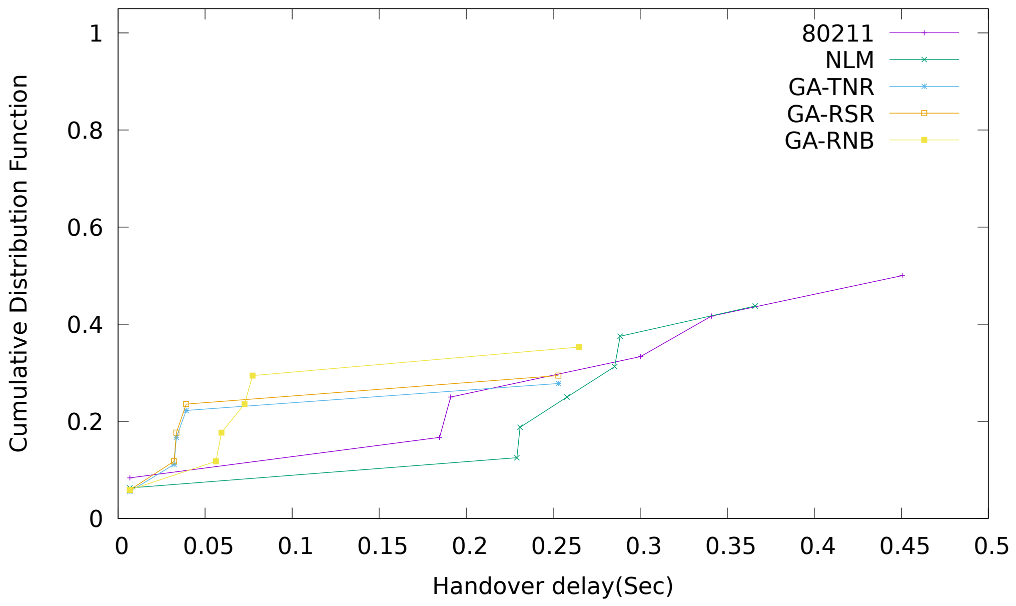


Figure 4.11: Handover delay vs. Cumulative distribution function when the STAs move randomly (Scenario II)

4.5.2.3 Changing the Number of APs

In Scenario III, one STA moves between different number of APs (6, 12, 18, 24, 30) at a constant speed of 3m/s. At the simulation start time $t = 0$, the STA is connected to AP0. The STA starts to move at simulation time $t = 0$. The simulation time is 60s for each run. This scenario is used to investigate the density impact on handover performance.

With the increasing number of APs, the handover frequencies increase and the STA has more chances to find new APs that can provide better throughput than the current serving AP. However, the expected communication time between the STA and AP is reduced because of the dense environment. The proposed self-optimizing approach affect throughput less than the 802.11 and NLM mechanisms because it considers the network conditions and dynamically adjusted scanning timers and scanning channels.

- **Throughput**

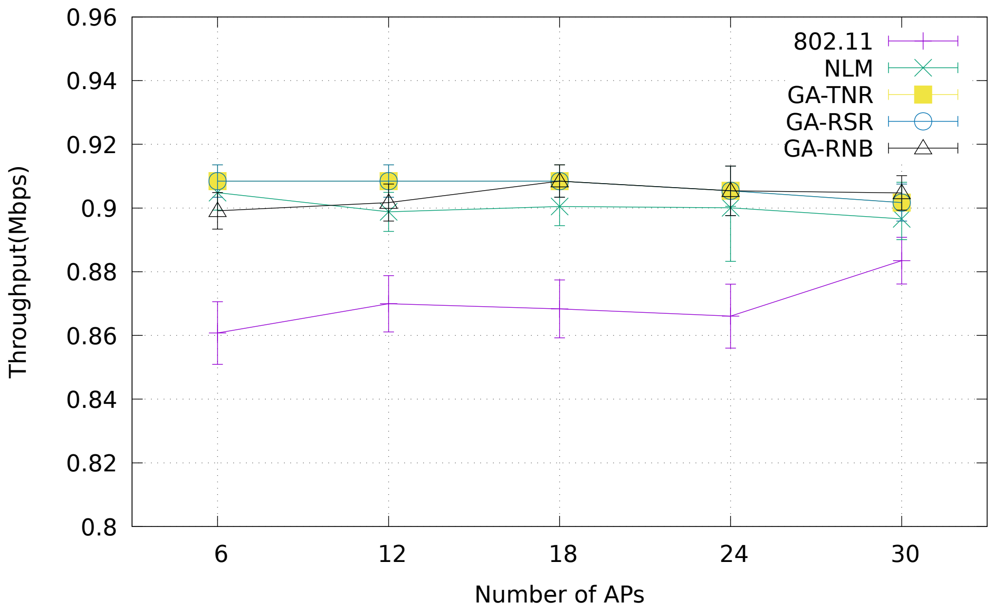


Figure 4.12: Throughput vs. Number of APs (Scenario III)

Figure 4.12 gives the average throughput among 802.11, NLM and the proposed self-optimizing method. The average throughput of the proposed methods is improved by 4.70% with GA-TNR, 4.70% with GA-RSR and 4.41% with GA-RNB compared with the 802.11 standard respectively. Compared with NLM, the average throughput of the proposed method is improved by 0.71% with GA-TNR, 0.71% with GA-RSR and 0.42% with GA-RNB.

The performance of GA-TNR and GA-RSR are almost the same in these density changing environments. When the number of AP increases, the improvements are also stable by using GA-TNR and GA-RSR. GA-TNR and GA-RSR achieve better throughput performance than GA-RNB because the boundary mutation in GA-RNB only uses upper or lower value to mutate the scanning parameter. This method cannot maintain the diversity of population and affect the efficiency to find better solutions of scanning parameters discussed in section 4.3.6.2. However, The GA-RNB performs better when the number of APs increased to 30 just because the rank-based selection is more robust towards optimal in a higher density environment.

- **Packet loss rate**

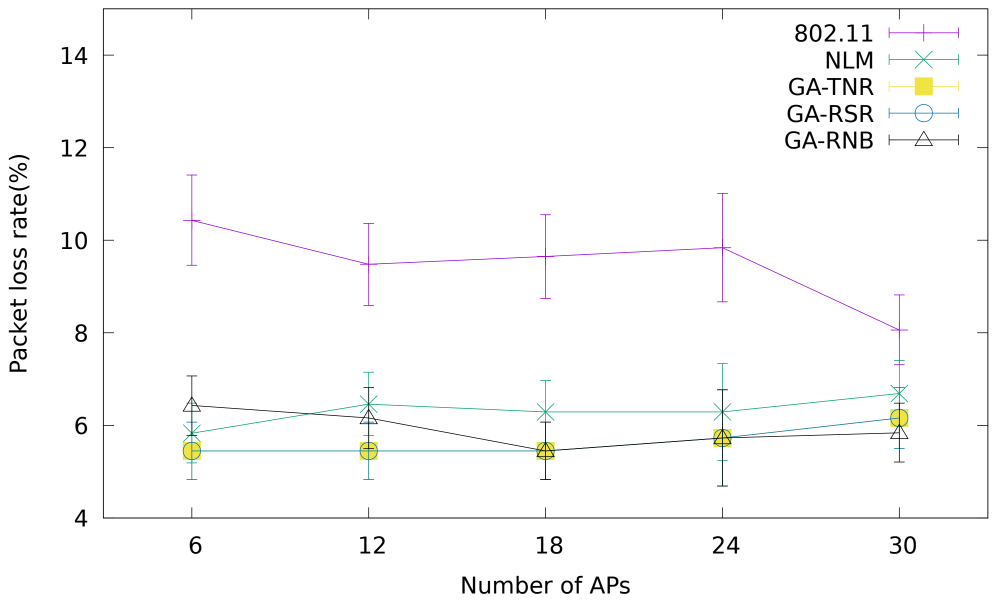


Figure 4.13: Packet loss rate vs. Number of APs (Scenario III)

The packet loss happens when handover occurs because of the disruption time. The packet loss rate also strongly depends on the density of networks. When the number of AP increases from six to twelve, the packet loss rate is reduced because the STA has more chances to associate with new APs that can provide better service than the current AP. However, when the number of AP increases, the packet loss rate increases a little because of the interferences between APs.

Compared with 802.11, the average packet loss rate of self-optimizing methods is reduced by 42.99% with GA-TNR, 42.99% with GA-RSR, 40.26% with GA-RNB. Although the average packet loss rate of NLM is reduced by 36.32%, the proposed approaches further reduce the packet loss rate by 10.46%, 10.46% and 6.18%, receptively. The GA-TNR and GA-RSR achieve the same improvement of reducing the average packet loss rate. This is because tournament selection used in GA-TNR and roulette wheel selection used in GA-RNB is faster than the ranked-based selection used in GA-RNB as shown in Table 2.6. Also, the random mutation used in both GA-TNR and GA-RSR can enhance the search efficiency to find better solutions than GA-RNB [83]. Therefore, the scanning parameters can be optimized faster to search for the best candidate AP to perform handover. The packet loss is reduced by reducing the disruption time caused by long handover delay.

- **Handover delay**

  Figure 4.14 shows the average handover delay impacted by different algorithms with varying the number of APs. As the number of AP increases, the total handover delay is not always increased. When the number of AP is increased to 12 and 18, the handover delay reduced because of the density of APs. The STA can easily find new APs to associate. However, when the number of AP is increased to 24, the handover delay starts to increase because of the interference between APs. Compared with standard 802.11 and NLM, the average handover delay of self-optimizing approaches is less affected by

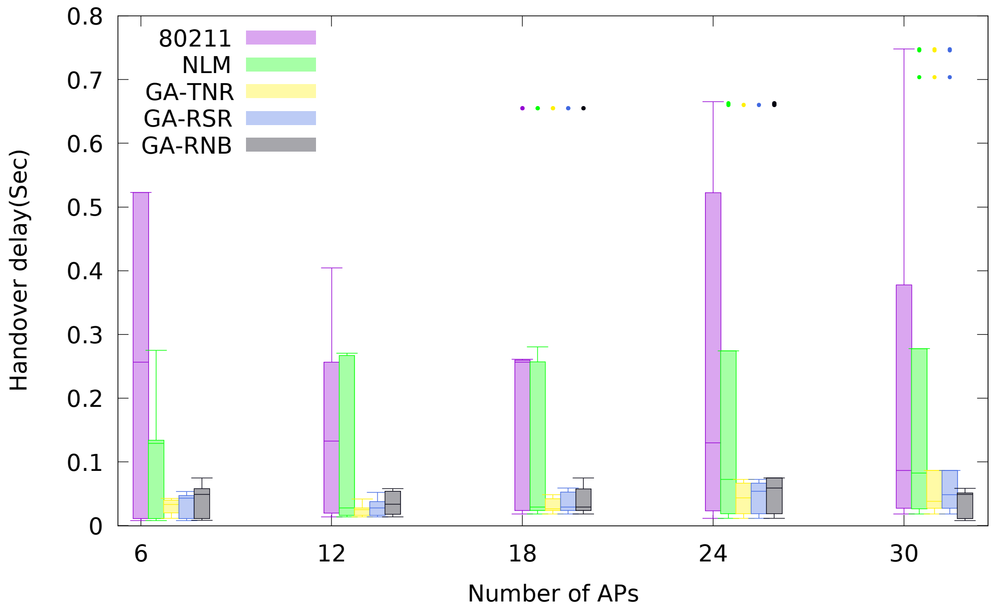


Figure 4.14: Handover delay vs. Number of APs (Scenario III)

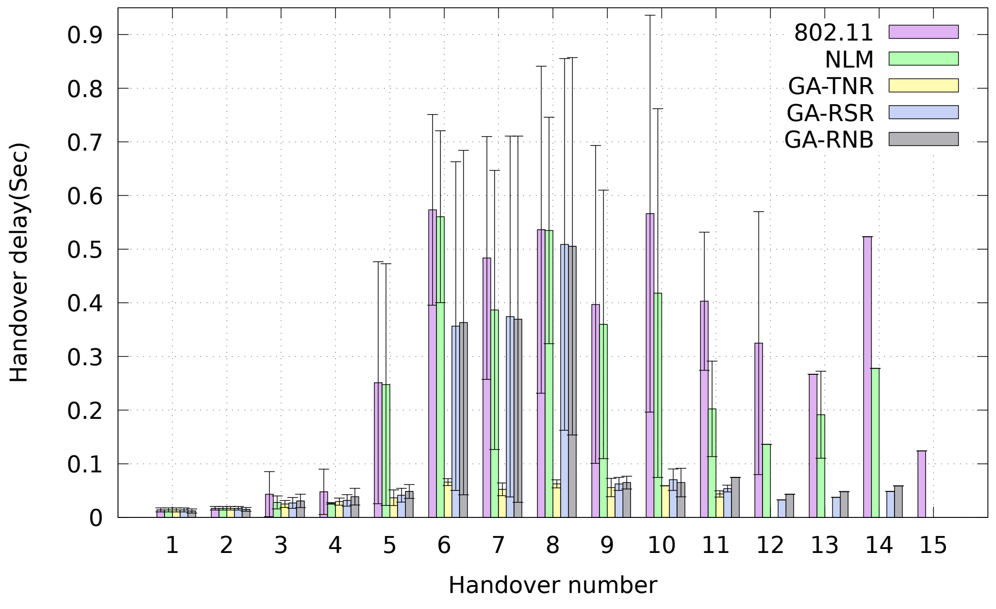


Figure 4.15: Handover delay vs. Handover number when the number of APs is changed (Scenario III)

the various density network. This is because the self-optimizing adjusts the scanning time according to the density change. Therefore, the total handover delay is reduced.

As can be seen in Figure 4.15, the average handover delay of the 802.11 standard is 0.30s/ho(handover number). The average handover delay of NLM is reduced to 0.25s/ho. However, the average handover delay of the self-optimizing approaches is reduced to 0.04s/ho with GA-TNR, 0.12s/ho with GA-RSR and 0.13s/ho with GA-RNB. The handover delay is significantly reduced by proposed algorithms because the adaptive scanning timers and channels are dynamically adjusted based on network conditions. This can reduce the scanning delay to make handover faster than the fixed value used in 802.11 standards. When the number of AP increases, GA-TNR method can converge faster to the optimal scanning parameters [64] than GA-RSR and GA-RNB because of the smallest computation time as shown in Table 2.6. Also, N-point crossover can cut the scanning parameters as a group per channel, which is suitable for this scanning optimization problem as discussed in section 4.3.5. Furthermore, random mutation performs better than boundary mutation used in GA-RNB. This is because the boundary mutation only chooses the upper or lower boundary value to replace the value of the gene. The diversity of the population is hard to be increased by GA-RNB and affect the efficiency of finding the best solution of scanning parameters [85]. Thus, the GA-RNB may not find the global optimal scanning parameters to reduce the scanning delay. The scanning delay causes the handover delay longer using GA-RNB than GA-TNR and GA-RSR.

4.5.2.4 Changing the Number of STAs

In Scenario IV, the number of STAs has been increased to see the impact on handover performance. The STAs move at t=0s with a constant speed of

3m/s from AP0 to AP5. The distance between APs is 30m. The simulation time is 60s for each run. This scenario is used to investigate the density impact on handover performance.

- **Throughput**

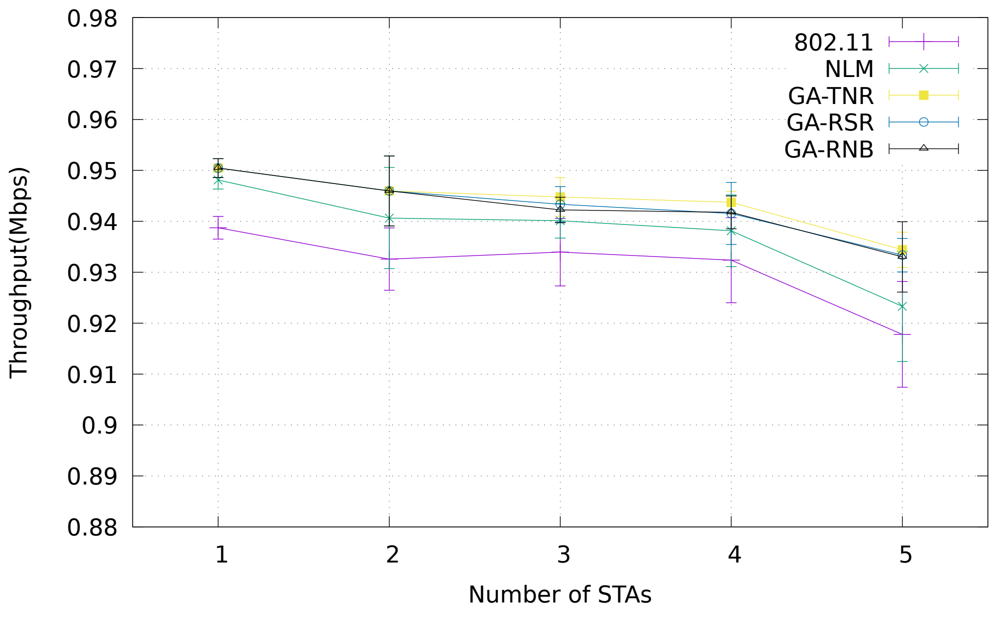


Figure 4.16: Throughput vs. Number of STAs (Scenario IV)

Figure. 4.16 shows the average STAs throughput impacted by different algorithms with various numbers of STAs. The average throughput of IEEE 802.11 drops 2.28% when the number of STAs changes from one to five, while the average throughput has been improved by around 0.75% using NLM method. This is because the NLM reduces the number of channels scanned. At the same time, the NLM considers the AP load balance. The STAs are prevented from handing over to overloaded APs. Thus, throughput performance is improved by NLM. However, NLM still uses fixed scanning parameters that cannot meet the dynamic network requirements. These fixed scanning parameters used by STAs can increase the scanning delay because the STAs spend the same time to scan each channel. The self-optimizing algorithms automatically adjust the scanning parameters based on real-time network environments such as channel conditions and AP load. Therefore, the average throughput has been further improved compared with NLM. Compared with the 802.11 standard, the average throughput has been improved by 1.37% with GA-TNR, 1.27% with GA-RSR and 1.25% with GA-RNB, respectively.

- **Packet loss rate**

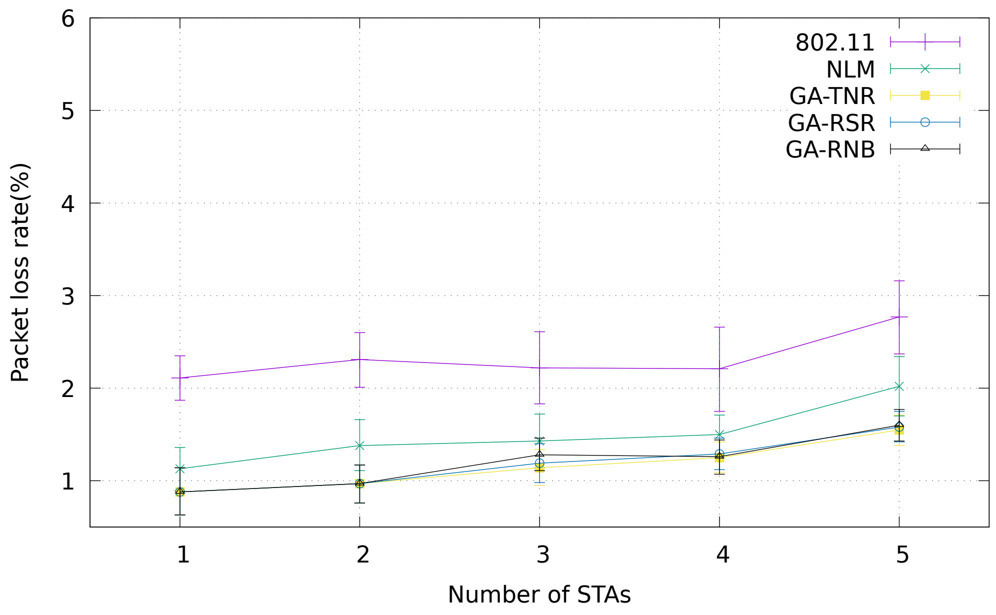


Figure 4.17: Packet loss rate vs. Number of STAs (Scenario IV)

Figure 4.17 illustrates the average STAs packet loss rate changes when the number of STAs increases. When the number of STAs is increasing, the contention among STAs increases. This contention causes a higher packet loss rate when the STAs become denser in the networks. Compared with 802.11, the average packet loss rate has been reduced by around 35.80% with the NLM method because NLM considers the packet loss rate, packet delay, and AP load when the neighbor list is updated. This NLM method prevents STAs from handing over to unnecessary AP and reduces the handover failure rate. Thus, the packet loss rate is reduced.

The packet loss rate has been further reduced by self-optimizing algorithms. The self-optimizing algorithms achieve more improvement than NLM because they not only reduce the number of channels but also optimize the scanning parameters. The packet loss rate compared with 802.11 has been reduced by 50.69% with GA-TNR, 49.14% with GA-RSR and 48.80% with GA-RNB respectively. With the self-optimizing algorithms, the collision among STAs has been reduced because the STAs do not have to re-associate with the same AP. The scanning parameters are dynamically adjusted based on different STAs QoS such as packet loss, packet delay, and throughput. Thus, the scanning time is different for each STA. This reduces the collision and contention among STAs. Thus, the overall packet loss rate has been improved more using self-optimizing algorithms than NLM.

- **Handover delay**

  Figure. 4.18 shows the average handover delay impacted by different algorithms with various numbers of STAs. As the number of STAs increases, the average handover delay of STAs increases because the density of STAs causes more collisions among STAs. This makes STAs spend more time scanning for the best candidate AP. Also, the STAs try to associate with the same AP without considering load balance because they start to move at the same time with the same speed and direction. This leads to load imbalance and causes the STAs to spend more time to find a new AP. However, compared with standard 802.11, the average handover delay of NLM and the self-optimizing approaches is reduced because the scanning list and parameters are updated based on network environment changes in-

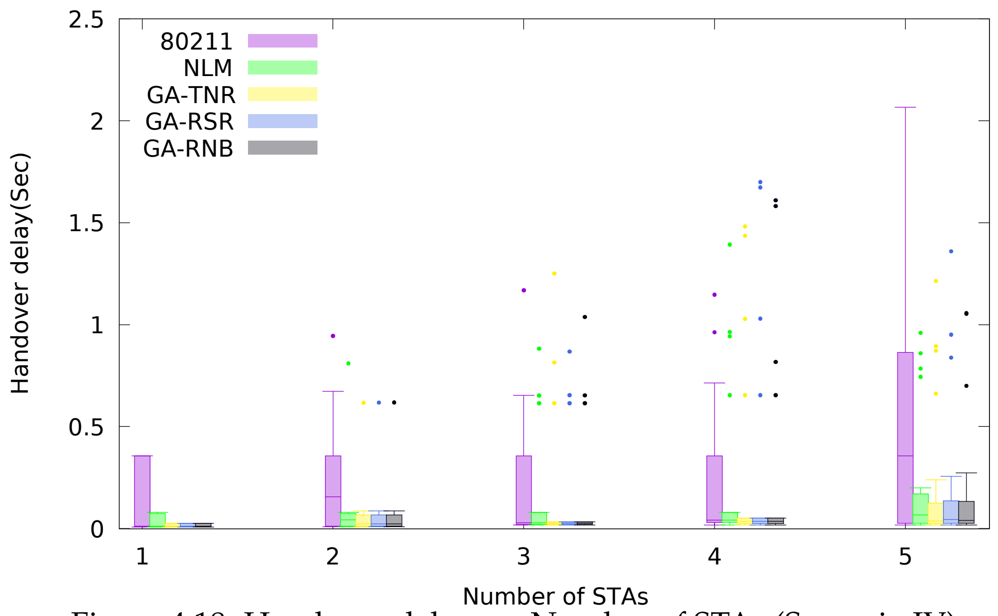


Figure 4.18: Handover delay vs. Number of STAs (Scenario IV)

cluding channel conditions, STA QoS and AP load. Thus, the average handover delay has been reduced by 51.06% with NLM, 74.47% with GA-TNR, 72.34% with GA-RSR and 70.21% with GA-RNB, respectively.

4.5.2.5 Changing the Network Topologies

In Scenario V, different topologies of AP placement are investigated. First, the APs are placed in one line as shown in Figure 4.19. The distance between two APs is 30m. One STA moves from AP0 to AP5 at t=0s with a constant speed of 3m/s from left to right backwards and forwards three times. The simulation time is 100s for each run.

Then, the APs are placed in grid as shown in Figure 4.20. The distance between the two lines is 16m and 12 APs are placed on each line. At t=0s, one STA moves in the middle among lines back and forth with a constant speed of 3m/s. The distance between APs is 60m.

Finally, Figure 4.21 shows the random AP placement (Poisson point

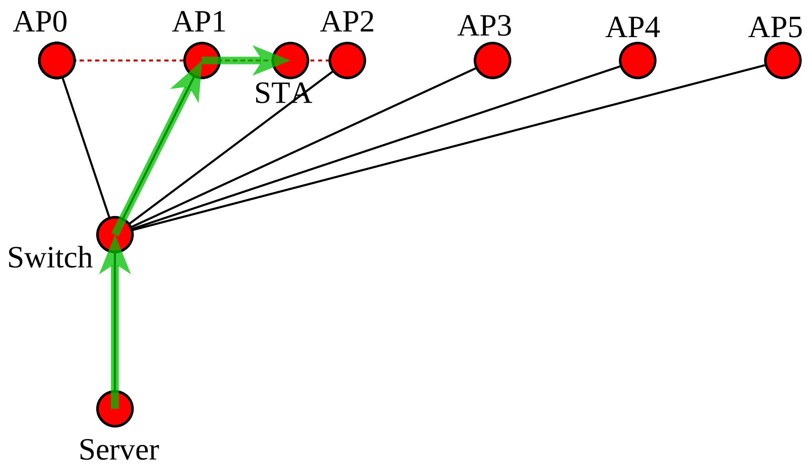


Figure 4.19: AP placement of 1D line topology for self-optimizing approach in NS3

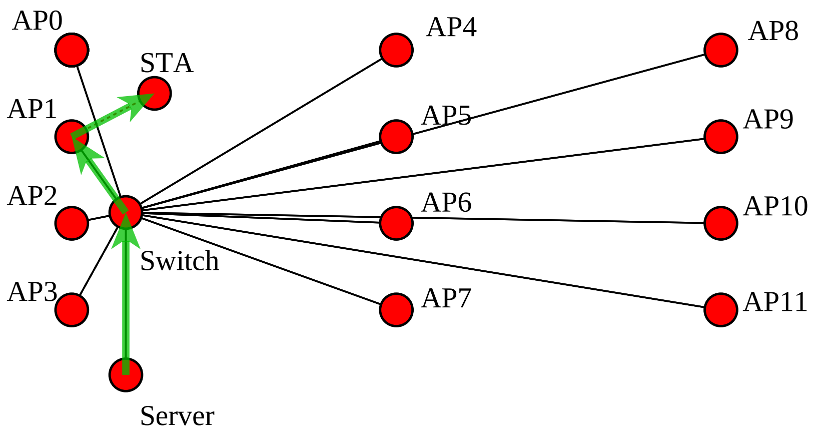


Figure 4.20: AP placement of 2D grid topology for self-optimizing approach in NS3

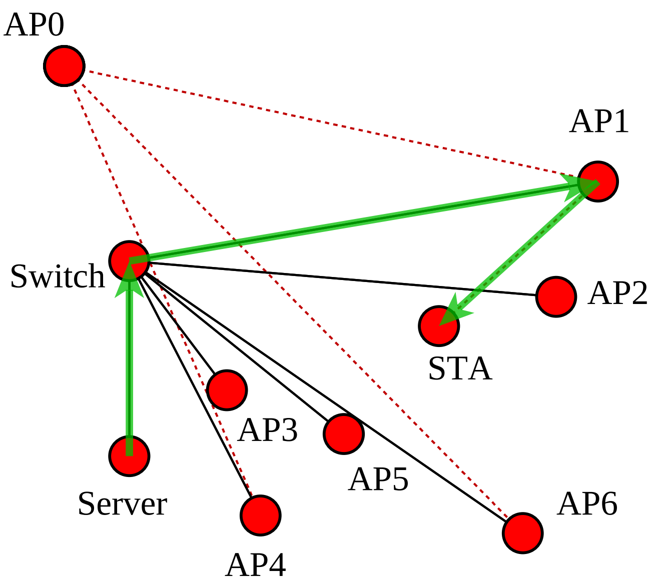


Figure 4.21: AP placement of random topology for self-optimizing approach in NS3

distribution) in a 90×80m full coverage area. One STA moves with a constant speed of 3m/s from left to right backwards and forwards.

- **Throughput**

  In Figure 4.22 4.23 4.24, the average throughput of self-optimizing approaches, NLM and 802.11 standard has been compared in three different network topologies.

  Figure 4.22 shows the average throughput changing when STA moves in a line topology. Compared with 802.11, the average throughput of the proposed self-optimizing method has been improved by 1.00% with NLM, 1.25% with GA-TNR, 1.00% with GA-RSR and 1.25% with GA-RNB, respectively. As this line topology is not dense, the number of neighbor APs is small for each AP. When a STA initiates a handover, the scanning time cannot be affected too much by the scanning parameters. Thus, the self-optimizing approaches achieve

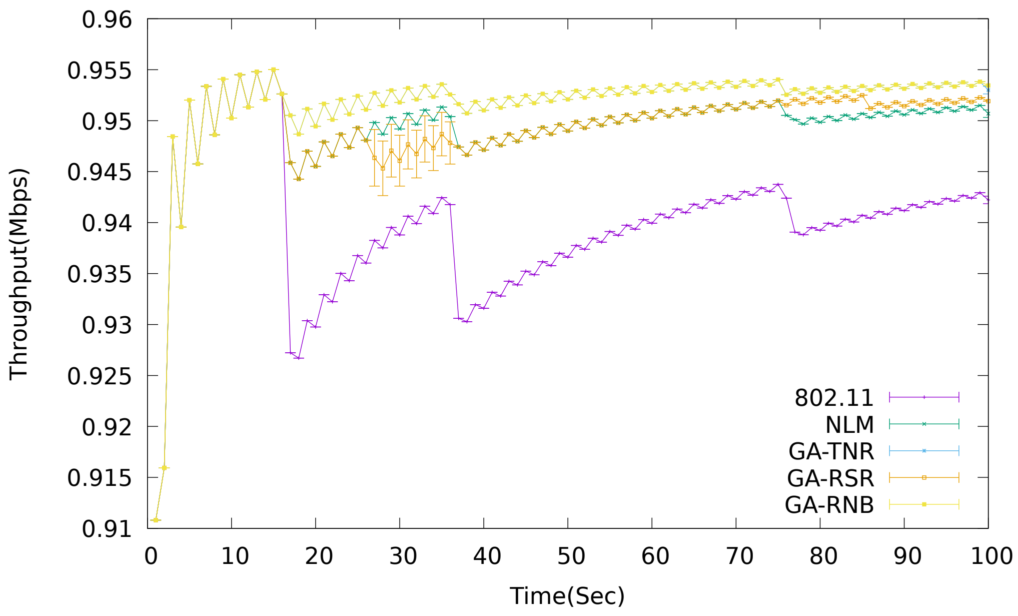


Figure 4.22: Throughput vs. Simulation time - 1D topology (Scenario V)

almost the same improvement of throughput as NLM that uses the fixed scanning parameters.

Figure 4.23 shows the average throughput changing when STA moves in a grid network topology. Compared with 802.11, the average throughput of the proposed self-optimizing method has been improved by 4.29% with NLM, 5.03% with GA-TNR, 4.81% with GA-RSR and 4.79% with GA-RNB, respectively. Compared with 1D network topology, the improvement of average throughput using the self-optimizing approach has been enhanced more in the 2D grid networks. This is because the AP density is increased in the 2D topology. More APs need to be scanned by a STA when the handover is initiated. The scanning time affects the throughput performance. With the GA algorithms, the scanning delay is reduced by the optimized scanning parameters that can meet the real-time network requirement. As GA-RNB algorithm uses ranked based selection, the searching time of optimal scanning parameters is much longer than GA-TNR and GA-RSR [129]. Thus, the performance of GA-RNB is worse than GA-TNR and GA-RSR. As the data size of scanning parameters is not big, GA-TNR using tournament selection can find the optimal scanning parameters very fast [127].

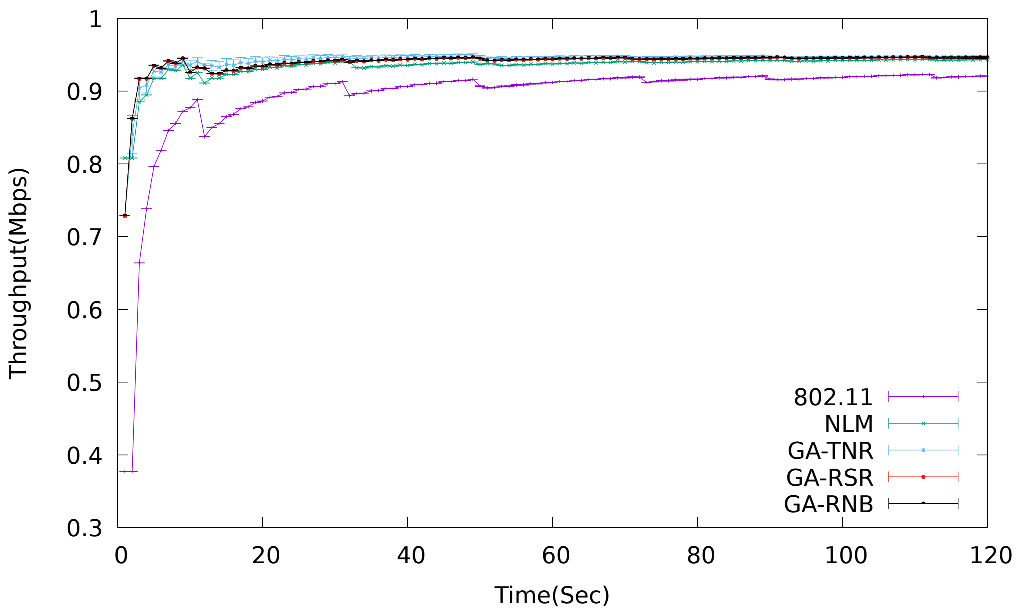


Figure 4.23: Throughput vs. Simulation time -2D topology (Scenario V)

Figure 4.24 shows the average throughput changing when STA moves in a random topology network. Compared with 802.11, the average throughput of the proposed self-optimizing method has been improved by 3.65% with NLM, 3.80% with GA-TNR, 3.70% with GA-RSR and 3.93% with GA-RNB, respectively. Compared with the 2D topology, the AP placement is less dense than the 2D topology in this random topology network. In this random topology, the improvement of GA-RNB is better than GA-TNR. This is because the individual scanning parameters in the population are sorted using ranked based selection in GA-RNB. The ranked based selection is more efficient at finding optimal scanning parameters in this less dense topology network. GA-TNR still outperforms GA-RSR because of fast convergence time. The faster the GA algorithms find the optimal value of scanning parameters, the lower the scanning delay is. Therefore, the average throughput can be improved by reducing the scanning delay to keep the data transfer continuously.

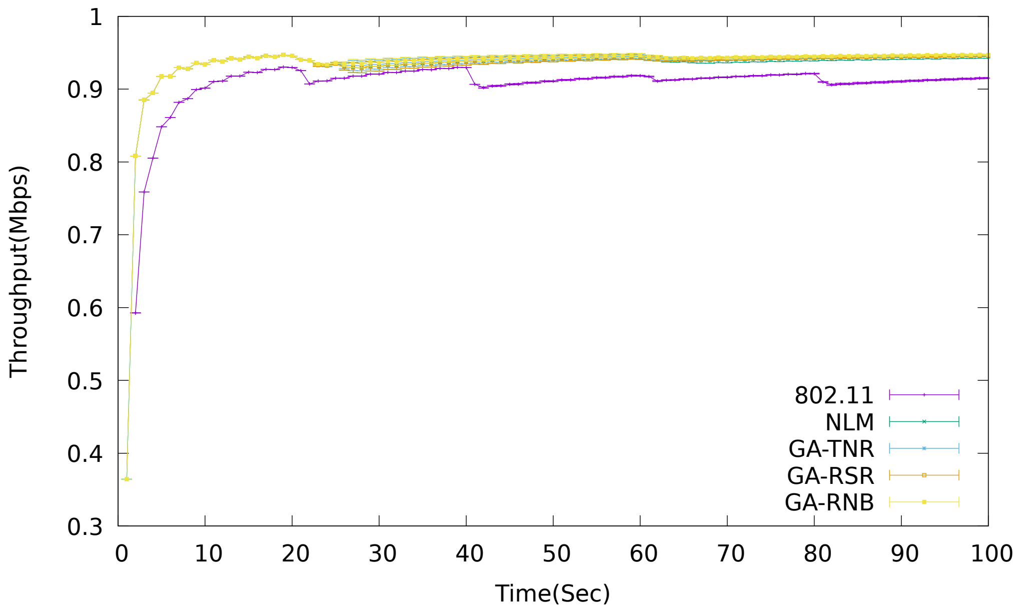


Figure 4.24: Throughput vs. Simulation time - Random topology (Scenario V)

- **Packet loss rate**

  In Figure 4.25 4.26 4.27, the packet loss rate of NLM approaches is compared with the 802.11 standard in three network topologies.

  Figure 4.25 shows the average packet loss rate changing when STA moves in a line network topology. Compared with 802.11, the average packet loss rate of self-optimizing approaches has been improved by 51.31% with NLM, 64.40% with GA-TNR, 51.31% with GA-RSR and 64.40% with GA-RNB, respectively. GA-TRN and GA-RNB show the same performance in this topology. This is because

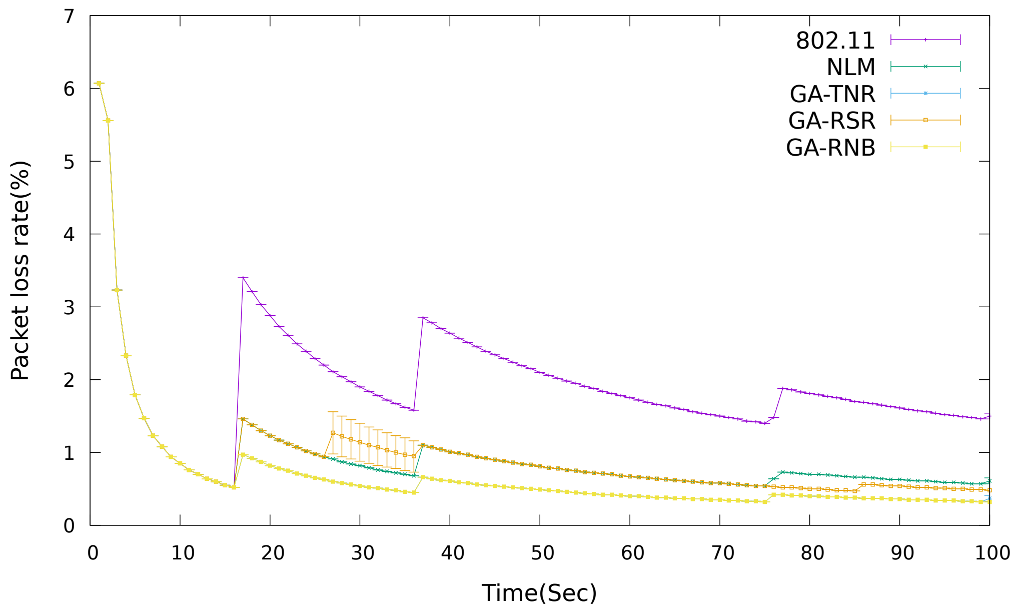


Figure 4.25: Packet loss rate vs. Simulation time - 1D topology (Scenario V)

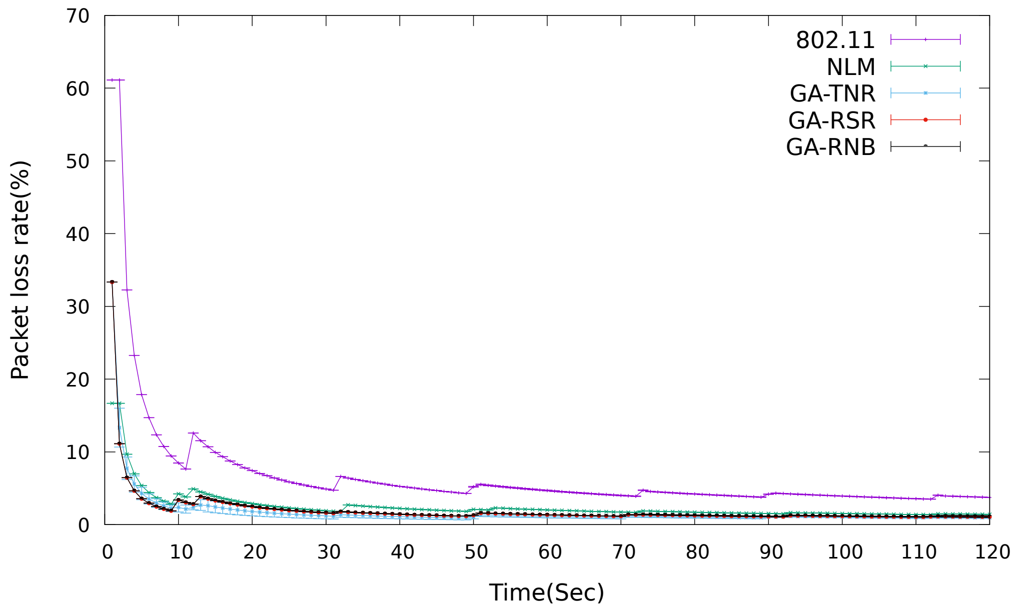


Figure 4.26: Packet loss rate vs. Simulation time - 2D topology (Scenario V)

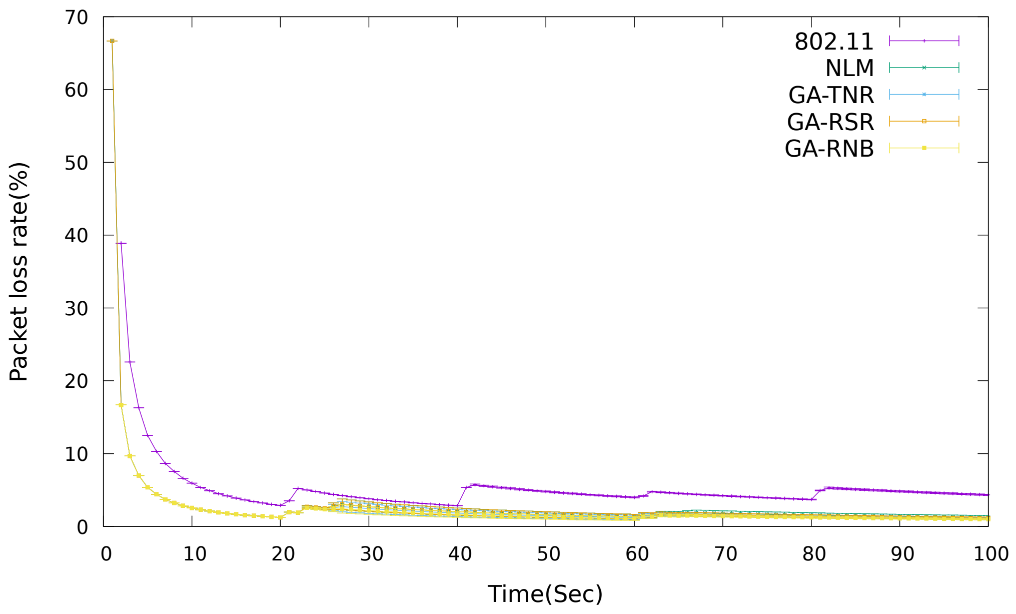


Figure 4.27: Packet loss rate vs. Simulation time - Random topology (Scenario V)

the tournament selection and ranked based selection is much more stable and faster than roulette wheel selection used in GA-RSR [73] [64]. These two algorithms can find the optimal scanning parameters quickly to reduce the scanning delay.

Figure 4.26 shows the average packet loss rate changing when the STA moves in a grid network topology. Compared with 802.11, the average packet loss rate of the proposed NLM method has been improved by 69.45% with NLM, 78.01% with GA-TNR, 74.46% with GA-RSR and 73.71% with GA-RNB, respectively. Compared with the 1D network topology, the packet loss rate of self-optimizing approach in 2D grid topology has been reduced more compared with the 802.11 standard and NLM. This is because the AP density is increased in the 2D topology. With an increasing number of APs, the interference is increased between APs. The scanning delay is also increased because of the dense AP placement. More APs need to be scanned by a STA when the handover is initiated. In traditional 802.11 networks, the data disruption time is increased because of frequency handover. With the GA algorithms, the scanning delay is reduced by the optimized scanning parameters and reduced number of channels. As GA-RNB algorithm uses ranked based selection, the searching time of optimal scanning parameters is much longer than GA-TNR and GA-RSR in the dense environment, the performance of GA-RNB is worse than GA-TNR and GA-RSR. GA-TNR achieves the best performance in the dense 2D networks because GA-TNR using tournament selection can converge very fast in this small data size optimization. The optimal scanning parameters of GA-TNR can be obtained fast enough to meet the dynamic network environments. Thus, the packet loss rate is reduced by more than GA-RSR and GA-RNB.

Figure 4.27 shows the average packet loss rate changing when STAs move in a random network topology. Compared with 802.11, the average packet loss rate of the proposed NLM method has been improved by 52.70% with NLM, 58.80% with GA-TNR, 55.28% with GA-RSR and 55.25% with GA-RNB, respectively. Compared with the 2D topology, the AP placement is less dense in this random topology network. The issue of interference between APs is reduced compared with 2D topology. Thus, the improvement of the packet loss rate is less than in the 2D topology. However, as the APs are randomly placed in the random topology, the distance between APs is not the same. The AP assignment to each channel is more randomly than 1D and 2D topologies. In the 802.11 standard, if one AP uses the same channel assigned to a neighbor AP, the network performance is degraded due to the channel interference [137]. In the self-optimizing approach, the channel is sorted based on the SNR, packet delay and AP load information. The STA will scan for the best candidate AP by the priority of channels. The best candidate

AP can be found earlier than the 802.11 standard. Thus, the scanning delay is reduced by the sequence channels with optimized scanning parameters.

- **Handover delay**

  Figure 4.28 illustrates the handover delay comparison among three different topologies.

  In 1D line topology, the average handover delay has been reduced by 0.12s/ho(handover numbers) with NLM, 0.15s/ho with GA-TNR, 0.14s/ho with GA-RSR, 0.15s/ho with GA-RNB compared with 802.11. NLM reduces less handover delay because it only reduces the number of channels scanned by a STA. With the self-optimizing algorithms, the scanning parameters and the number of channels are all optimized in order to reduce the scanning delay. GA-TNR and GA-RNB show the same improvement in handover delay. This is because the N-point crossover has a wide search space than single-point crossover. Thus, GA-TNR and GA-RNB can find better solutions for scanning parameters than GA-RSR. These scanning parameters can meet the network environments to reduce the scanning delay.

  Compared with the 1D topology, the APs are placed very densely in a 2D grid topology. The STA needs to scan longer to select the best candidate AP among the dense APs. The scanning delay increases the handover delay caused by the density. As the self-optimizing algorithms consider network conditions such as load, SNR and packet delay, the scanning parameters are dynamically adjusted to suit the network requirements. Thus, the scanning delay is reduced and the handover success rate has been increased. The average handover delay has been reduced by self-optimizing algorithms. The handover delay has been reduced more than NLM because NLM only reduces the number of channels and APs scanned by a STA. Self-optimizing algorithms optimized both the sequence of channels and scanning timers. Therefore, the results are better than NLM.

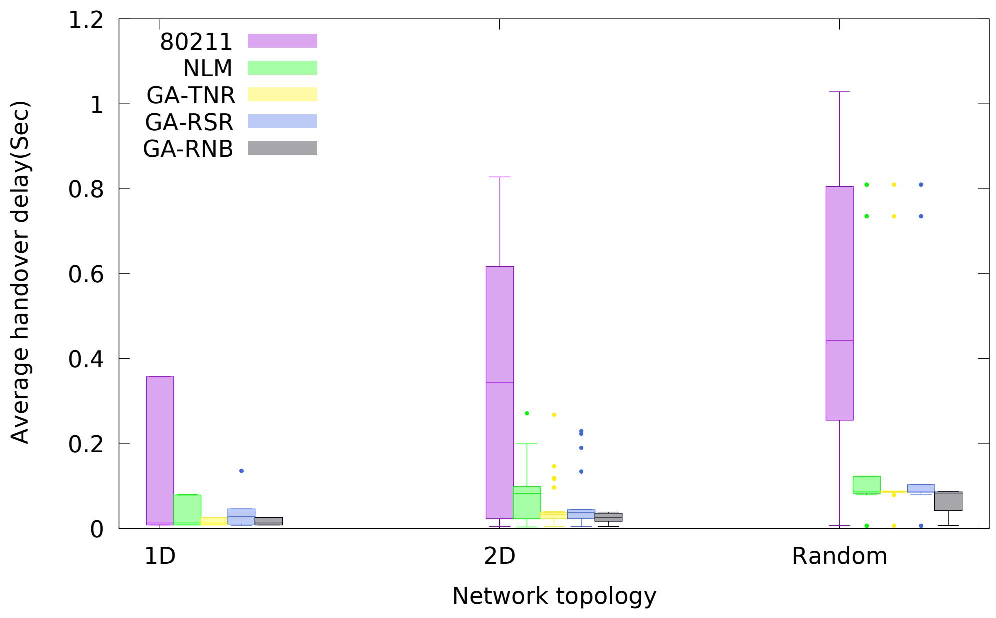


Figure 4.28: Handover delay in different network topologies (Scenario V)

In the random topology, the average handover delay has been reduced by 0.22s/ho(handover numbers) with NLM, 0.28s/ho with GA-TNR, 0.25s/ho with GA-RSR, 0.23s/ho with GA-RNB compared with 802.11. The main difference among random topology, 2D and 1D topology is that random AP placement causes different distances between APs. However, the scanning time that STA has to wait for the probe responses from each AP on each channel is the same in traditional 802.11 networks. This same scanning time can generate wasted waiting time if there is no AP on a channel or can miss the best AP if the scanning time is very small. In this random topology, the scanning time is even more affected than 1D and 2D topology because of varying distances between APs. The scanning time that is affected by the probe waiting time has to be dynamically adjusted by the adaptive scanning parameters. With the self-optimizing algorithms, the scanning parameters have been optimized based on the network environment. Therefore, the average handover delay is reduced compared with 802.11 and NLM.

4.5.3 GA Performance Evaluation

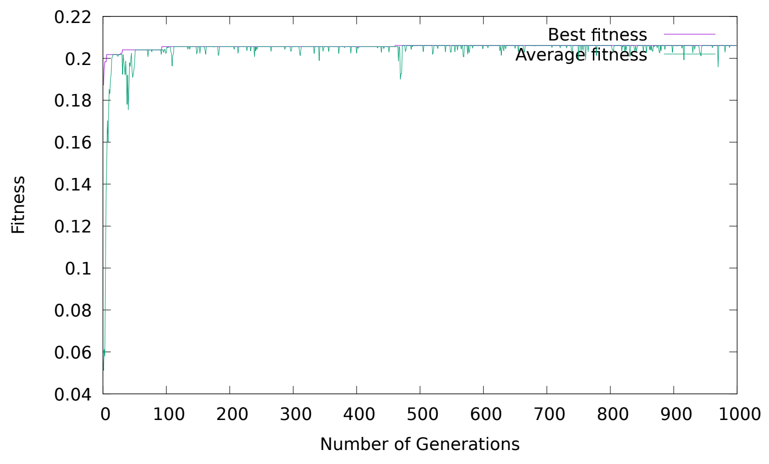
In this section, different GA selection, crossover and mutation operators are investigated for solving scanning parameters optimization problem in WiFi networks.

4.5.3.1 GA Selection Performance

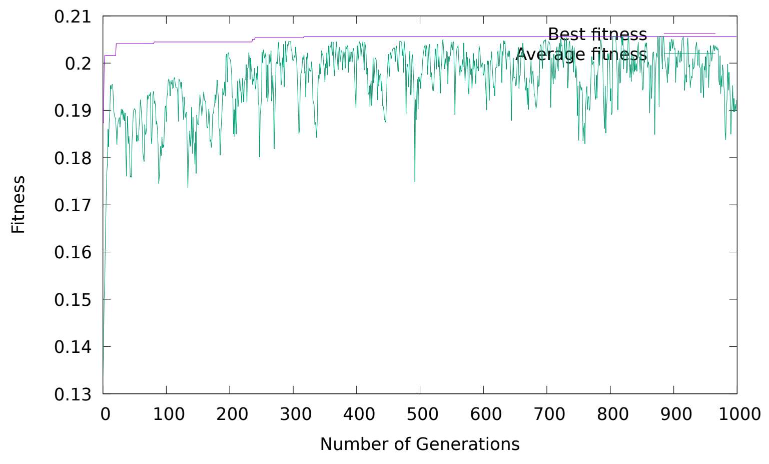
In order to evaluate the selection operators performance, the crossover operator and mutation operator are fixed. The simple point crossover and random mutation are chosen to investigate different selection methods.

In Figure 4.29a 4.29b 4.29c, the average fitness and best fitness are compared to investigate GA performance. The average fitness is the mean fitness value across the entire population. In each generation, the population changes and will get a new average population fitness. The best fitness tends to be improved as the iterations increase and the average fitness is always inferior to the best fitness, but the difference between them narrows at different generations in these three selection operators. This plateau occurs because the algorithm finds better and better solutions that are harder to improve upon.

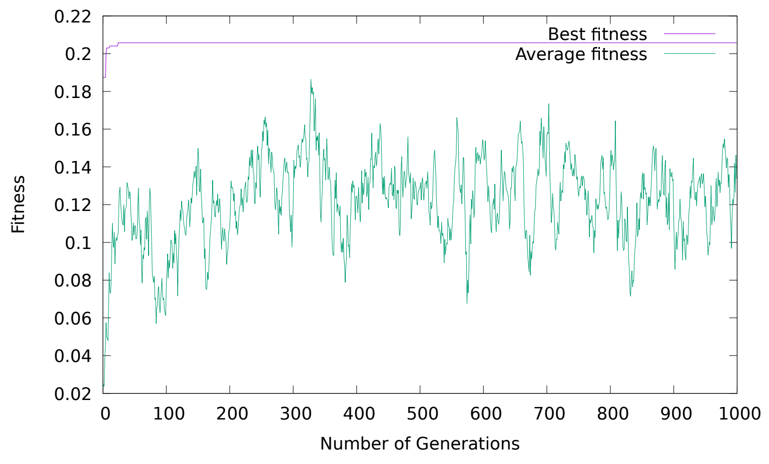
In Figure 4.29a, the difference between the average fitness and best fitness narrows to a constant after 110 generations. The convergence using roulette wheel selection in Figure 4.29b is slower than tournament selection. The difference between the average fitness and best fitness narrows after 300 generations. In Figure 4.29c, there is always a big difference between the average fitness and best fitness during 1000 generations. As the data size of this scanning parameters optimization problem is small, tournament selection converges very fast in the small size problem [127].



(a) Tournament selection



(b) Roulette Wheel Selection



(c) Ranked-base selection

Figure 4.29: The mean fitness and best fitness of the different GA selections

Therefore, the tournament selection has the best GA performance among these three operators.

As seen in Figure 4.30, as the number of generations increases, the trend of the best fitness value goes higher. In the beginning, the best fitness increases rapidly and then plateaus at 110 generations for tournament selection, 300 generations for roulette wheel selection and 50 generations for ranked based selection, respectively. Compared with roulette wheel selection and ranked based selection, tournament selection reaches the best fitness value of 0.2060.

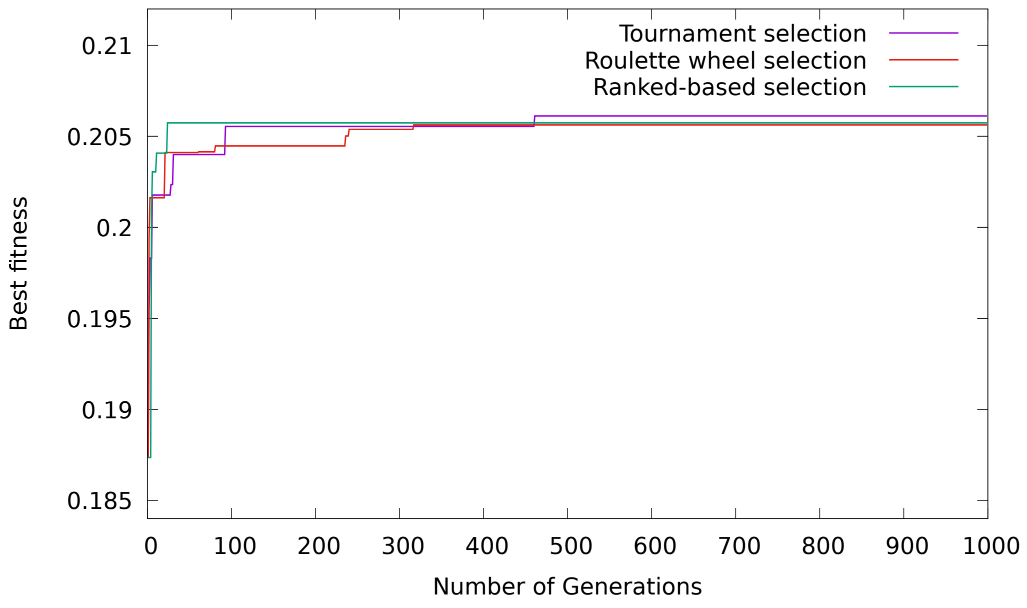
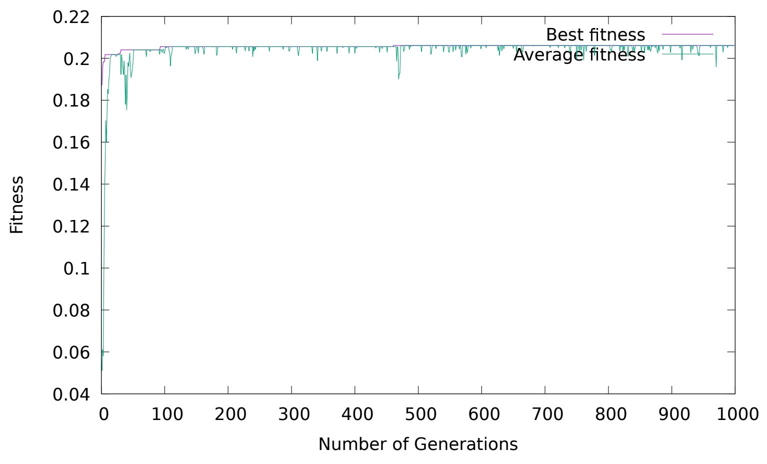


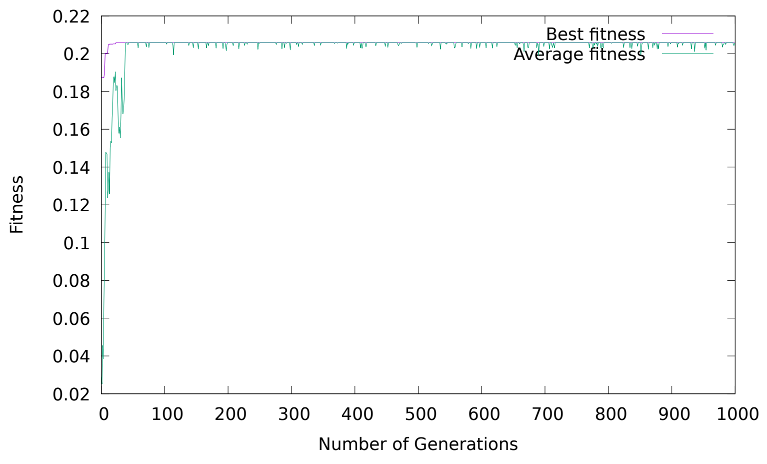
Figure 4.30: The best fitness of different GA selections with increasing generations

4.5.3.2 GA Crossover Performance

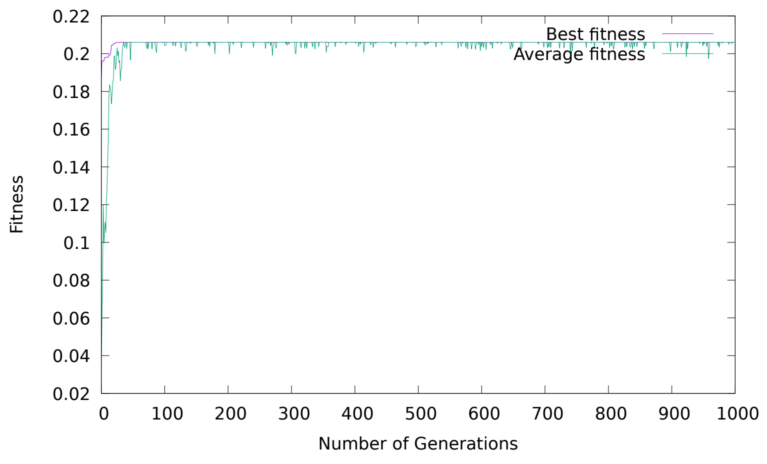
As tournament selection is the best, the tournament selection and random mutation are fixed to evaluate different crossover operators performance. Three crossover operators are compared by the best fitness and average



(a) Single-point crossover



(b) N-point crossover



(c) Uniform crossover

Figure 4.31: The mean fitness and best fitness of the different GA crossover operators

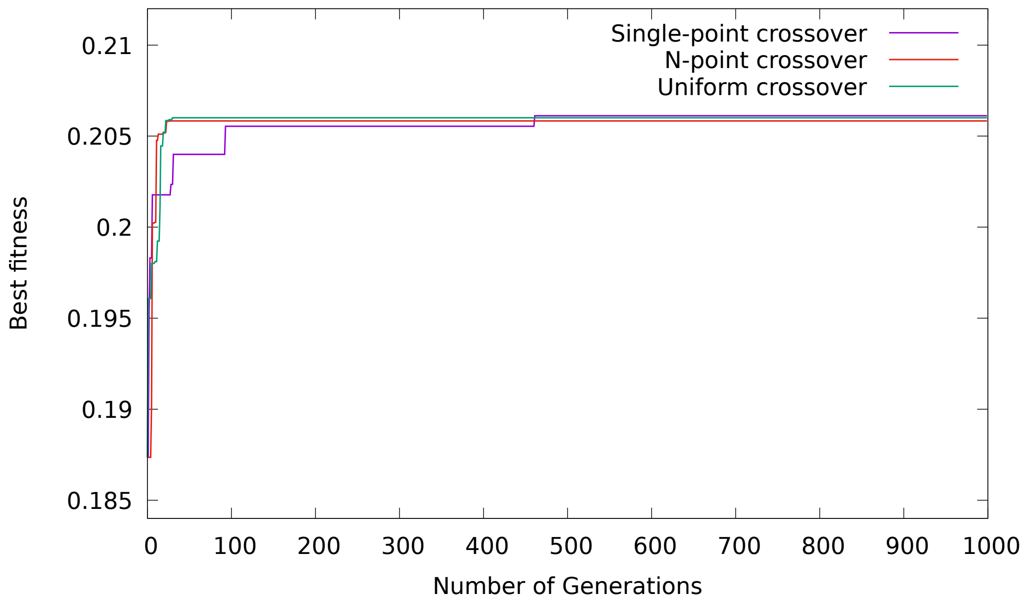


Figure 4.32: The best fitness of different GA crossover operators with increasing generations

fitness values. These are single-point crossover, N-point crossover and uniform crossover.

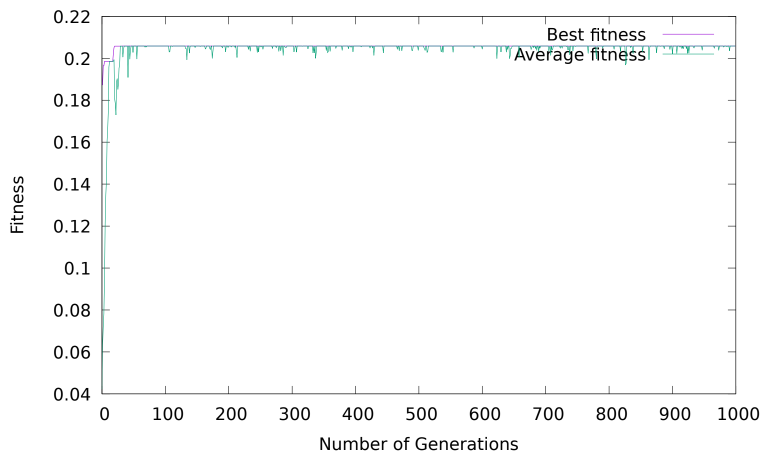
In Figure 4.31a 4.31b 4.31c, the average fitness and best fitness are compared to investigate GA performance. In Figure 4.31a, the difference between the average fitness and best fitness narrows to a constant after 112 generations. In Figure 4.31b and 4.31c, the convergences using N-point crossover and uniform crossover are faster than single-point crossover. The difference between the average fitness and best fitness narrows after 42 generations and 41 generations, respectively. As single-point crossover only selects one cross point to combine the parents to produce a new child chromosome, it is harder to move out of the local optimum. N-point can randomly select more crossover points than single-point and uniform crossover can also uniformly combine two parents. Both of them have a wide search space that can find global optimal scanning parameters for this optimization problem. Therefore, the N-point crossover and uniform crossover are more suitable for the RVC GA applied in this self-optimizing approach.

Figure 4.32 shows the fitness value trend when the number of generations increases. After around 40 generations, both N-point crossover and uniform crossover converge to a constant value of 0.2058 and 0.2060. However, single-point crossover needs to wait until 473 generations to reach the best fitness value. As discussed in section 4.3.5, the scanning parameters $t_{min}(i)$, $t_{max}(i)$, $T_{P_{req}}(i)$ and $T_{sw}(i)$ in the proposed GA method are considered as a group of parameters per channel. Therefore, the N-points crossover method is suitable to use for cutting the random position based on channel numbers.

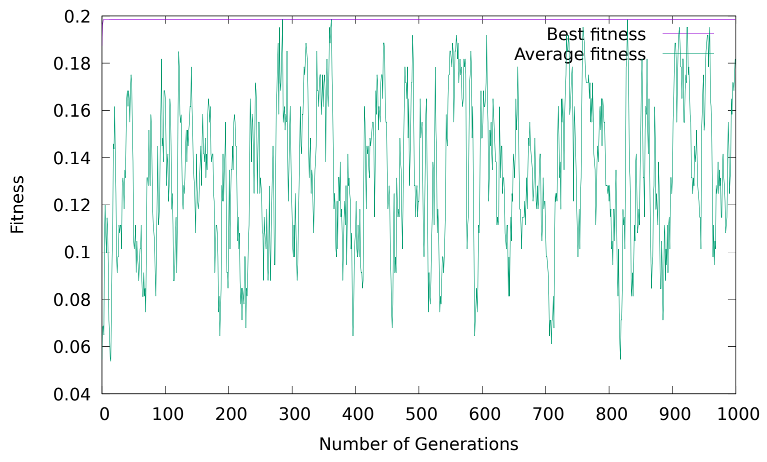
4.5.3.3 GA Mutation Performance

As tournament selection and N-point crossover are investigated to be used for solving the scanning parameters problems, these are chosen as fixed selection and crossover operators to evaluate crossover performance. Three mutation operators including random mutation, boundary mutation and non-uniform mutation are compared by best fitness and average fitness values.

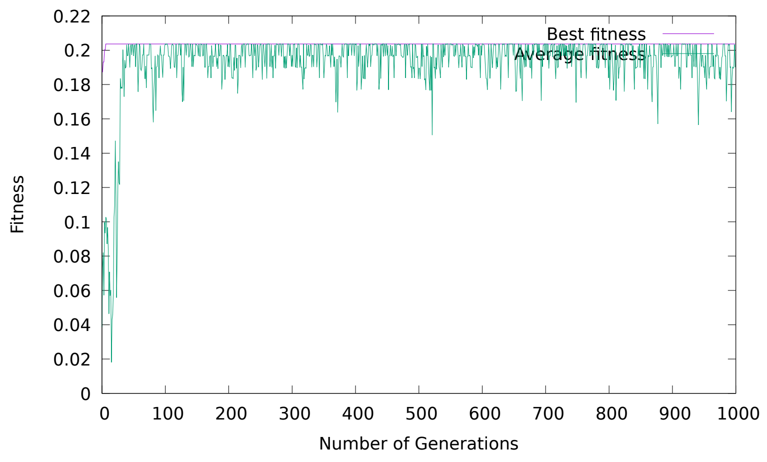
In Figure 4.33a 4.33b 4.33c, the average fitness and best fitness are compared to investigate GA performance. The performance of random mutation shown in Figure 4.33a is the best of these three mutation operations. In random mutation, the difference between the average fitness and best fitness narrows to a constant after 46 generations. Random mutation is better than boundary and non-uniform mutation because random mutation replaces the value of a gene with a random value(uniform probability distribution). This can ensure the GA can search the solution space freely. However, for the boundary mutation shown in Figure 4.33b, there is always a big difference between the average fitness and best fitness during the whole GA process. The average fitness value and the best fitness value cannot converge with an optimal solution. As a gene represents a scanning



(a) Random mutation



(b) Boundary mutation



(c) Non-uniform mutation

Figure 4.33: The mean fitness and best fitness of the different GA mutation operators

parameter, the boundary mutation mutes the genes only by the upper or lower boundary value. The diversity of scanning parameters cannot be satisfied by the boundary mutation. Thus, the boundary mutation cannot find an optimal solution for the scanning parameters. The convergence using non-uniform mutation in Figure 4.33c is slower than random mutation. This is because the value used to replace the gene need to be tuned. Compared with random mutation, the computation time is longer. Thus, the convergence is slower than non-uniform mutation.

As seen in Figure 4.34, the random mutation can obtain the best fitness value of 0.2060. The best fitness value of non-uniform mutation is 0.204. The boundary mutation has the lowest best fitness value of 0.1985. Therefore, the random mutation is chosen in the RVC GA for solving the scanning parameter optimization problem.

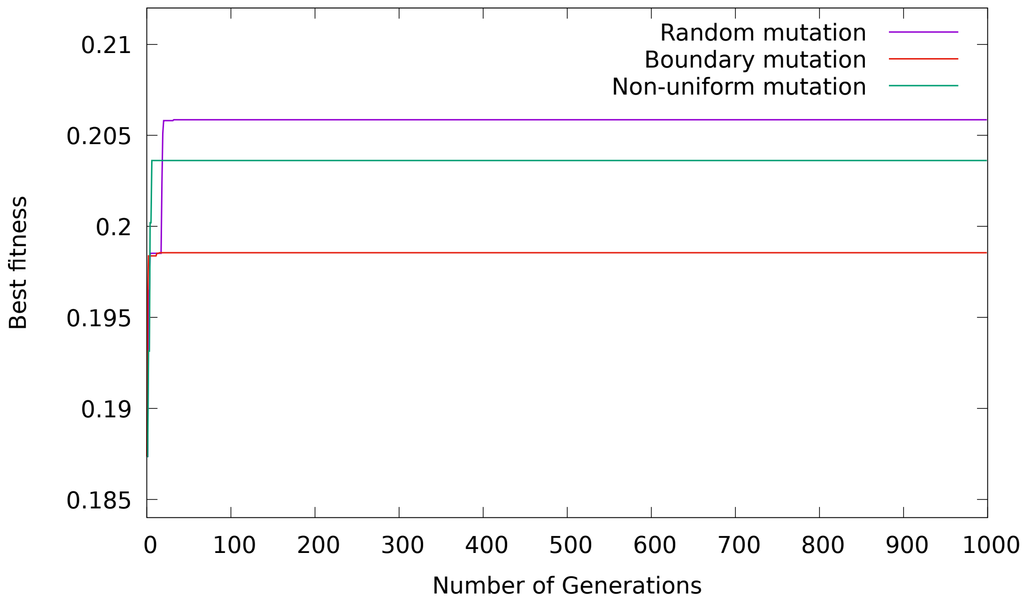


Figure 4.34: The best fitness of different GA mutation operators with increasing generations

4.6 Summary

The proposed self-optimizing approach based on Genetic Algorithms optimized the scanning parameters dynamically to adapt to the changes in the network environment. Due to the adaptive scanning parameters and reduction of scanning channels, the overall handover performance has been improved compared with 802.11 standard and NLM.

As the self-optimizing using GA only consider the candidate AP priority based on SNR, other radio parameters such as RSSI, delay and load can be considered to evaluate the self-optimizing approach in future. Moreover, the crossover rate and mutation rate have fixed values in this thesis. These two parameters can be optimized dynamically based on different data size and GA generations to get more optimal scanning parameters.

Chapter 5

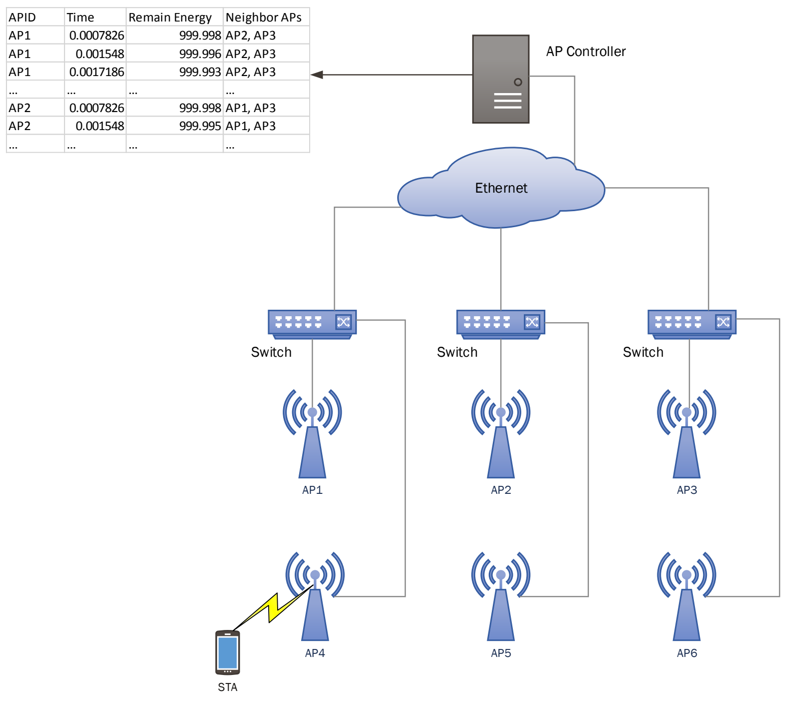
Self-healing System

Chapters 3 and 4 described how the scanning delay has been reduced by the self-configuring neighbor lists and self-optimizing algorithms, respectively. The study shows that the best candidate AP can be found within a short scanning time, improving handover performance when compared with traditional 802.11 networks. However, the self-configuring and self-optimizing algorithms only focus on the scanning phase. Even though the best AP is found, the STA cannot be prevented from associating with an AP that suddenly cannot work during the association phase. This will cause the handover failure and degrade the network performance. Also, the STAs associated with the faulty APs cannot be detected by self-configuring neighbor lists and self-optimizing algorithms. Therefore, fault management is needed to automatically monitor the network elements and compensate for the network degradation when faults occur.

A self-healing approach for WiFi networks is proposed to solve the unexpected outage issues described in this chapter. In WiFi networks, self-healing is divided into two parts: self-detection and self-compensation. Self-detection aims to automatically detect the faulty APs in dynamic networks. After the self-detection phase, the STAs associated with a faulty AP will handover to the neighbor APs. The neighbor APs' information can be obtained by the neighbor list discussed in Chapter 3. At the same time, the neighbor list will be updated and the scanning parameters will be optimized. Therefore, the three Self-Organizing Networks (SON) functionalities can work together to improve the overall network performance for WiFi networks.

5.1 Self-healing System

In this section, the proposed self-healing approach is introduced for WiFi networks. The proposed approach uses the ideas from Software Defined Networking (SDN) by using an AP controller (APC) to collect the measurement data from APs attached to the APC. The APC uses the OpenFlow 1.3 protocol [138] to communicate with individual APs. The self-healing network architecture is shown in Figure 5.1



APID	Time	Remain Energy	Neighbor APs
AP1	0.0007826	999.998	AP2, AP3
AP1	0.001548	999.996	AP2, AP3
AP1	0.0017186	999.993	AP2, AP3
...	...	...	...
AP2	0.0007826	999.998	AP1, AP3
AP2	0.001548	999.995	AP1, AP3
...	...	...	...

Figure 5.1: Self-healing architecture including an AP controller (APC) containing measurement dataset

Measured AP data is collected periodically and recorded in a table and stored in the APC. The neighbor APs information on each AP, which can be obtained based on the "Neighbor Report" mechanism specified in the IEEE 802.11k [136, 139], is saved in the table. The machine learning algorithms are triggered to detect the faulty APs periodically. Once the faulty APs are found, this information will be sent to STAs. The STAs are prevented from initiating a handover to faulty APs. The STAs that are associated with the faulty APs will be forced to handover to the neighbor APs promptly. Each STA in the vicinity will drop the faulty AP from its neighbor list until the AP can operate in its normal state.

5.1.1 Self-healing Framework

Self-healing is a functionality aiming to minimize the network performance deterioration when failures occur in a network element such as an AP. This self-healing approach includes detecting faulty APs and compensating for network degradation caused by faulty APs. Figure 5.2 shows the self-healing framework which consists of measurement collection, a self-healing process and performance evaluation.

5.1.1.1 Measuring AP Health

As discussed in section 3.1, the IEEE 802.11k-2008 standard provides mechanisms for gathering data on radio link performance and on the radio environment (called radio measurement reports). In the self-healing approach, the radio measurement reports are still collected for measuring the AP status. These measurement reports include radio and network information such as RSSI, load information, neighbor report, delay information, power constraints, power capability, Transmit Power Control (TPC) report, Received Channel Power Indicator (RCPI) to facilitate the management and maintenance of a Wireless Local Area Network (WLAN).

In this section, the collected measurement data including TPC report,

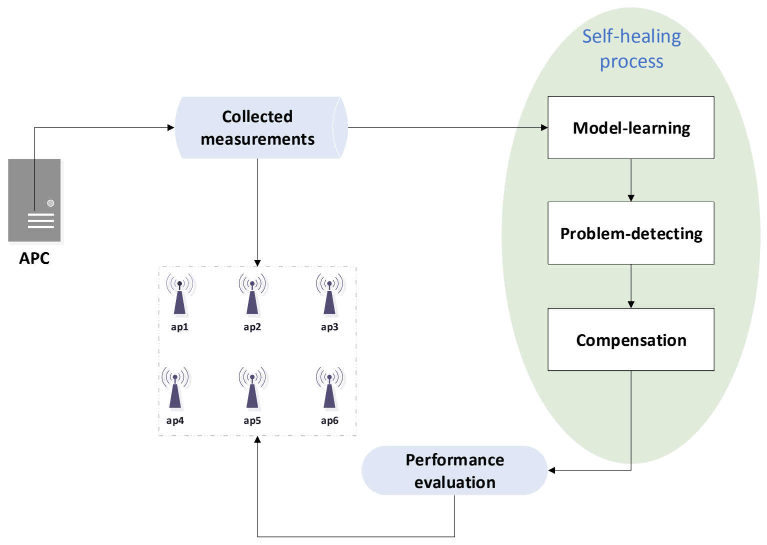


Figure 5.2: Self-healing process by using machine learning algorithms

Table 5.1: AP-level data for self-healing

Measurements	Description	Unit
TTI	Transmission Time Interval	10ms
AP Identification (APID)	BSSID	Integer
Energy measurement	The remaining energy of each AP	Joule (J)
Neighbor APs information	Refer to Table 3.1	Unit of Table 3.1

RCPI, power constraints and power capability are used to monitor the AP energy consumptions and the AP status is detected using machine learning algorithms based on these collected measurements.

The required AP-level data in a self-healing system is shown in Table 5.1. The measurement data is measured by individual STAs and sent to the APC via the APs. This happens as part of the standard 802.11 association process (called the probing phase). The proposed framework takes the measured data and sends it to the APC via custom OpenFlow messages (called experimenter fields in OpenFlow 1.3 [138]). When the APC obtains the measured data, the neighbor list will be updated with the removal of faulty APs. The STAs use the optimized scanning parameters to find the best AP to associate with in order to compensate for network degradation.

5.1.1.2 Self-detection

Since the data is collected from different APs at different times, the time sequences are sliced to sub-sequences and the data of each AP is selected in the same sub-sequences. These data sequences are used to find the faulty AP time and correctly identify a faulty AP (via its APID). The details are given in section 5.2.

5.1.1.3 Self-compensation

After the self-detection phase, if a faulty AP is found, the APC will send the neighbor AP's information to the STAs associated with the faulty APs. As discussed in chapter 3, the APs on the neighbor lists are updated based on the real-time network parameters shown in Table 3.1. Thus, the neighbor list can be updated when a faulty AP is detected and the STAs are prevented from associating with a faulty AP. With the neighbor APs' information, the STAs associated with a faulty AP will also be forced to handover to the neighbor APs. Therefore, the neighbor list is used for network compensation when the faulty AP is detected.

The ratified 802.11k amendment makes it possible for an AP to inform the STAs how to use TPC capabilities to change their transmit amplitude dynamically to match the APs power. For the faulty APs, they will be deleted from the neighbor list until the AP is restored, which is also a self-configuration function for neighbor list. Note that the 802.11k amendment [139] was incorporated into the IEEE 802.11 standard in 2012.

5.2 Problem Formulation

In this section, a detailed description of the faulty AP detection algorithms is presented. Self-detection aims to determine the existence of faulty APs and allow the APC to handover on-going connections to non-faulty APs in the vicinity. This self-detection improves the QoS through minimising throughput degradation and packet loss during a handover.

Let $A_j(ap)$ denote a set of APs in the network on the jth TTI and $X_j(n)$ denote the measured data of each AP whereby AP is indexed by j such that $1 \leq j \leq n$ and $j \in \mathbb{Z}^+$. The sets $A_j(ap)$ and measurement data $X_j(n)$ are expressed as follows:

$$A_j(ap) = \{ap_{j,1}, ..ap_{j,k}, ...ap_{j,n}\}, ap_{j,k} \in \mathbb{N}, \tag{5.1}$$

$$X_j(n) = \{x_{j,1}, ..x_{j,k}, ...x_{j,n}\}, \tag{5.2}$$

where $ap_{j,k}$ denotes the AP on the jth TTI with index k and $\mathbb{N}$ is natural numbers; $x_{j,k}$ denotes the measurement data of $ap_{j,k}$ and $x_{j,k}$ is a n-tuple defined as:

$$x_{j,k} = \{energy_s, energy_{n_1}, ..., energy_{n_m}\}, \tag{5.3}$$

where $energy_s$ denotes the remaining energy of the $ap_{j,k}$ and $energy_{n_1}$ denotes the remaining energy of the first neighbor AP of $ap_{j,k}$ on the neighbor list, and the number of neighbor AP is m. The neighbor list can be obtained by APC as described in section 3.2.

Let $f(x_{j,k})$ denote a learned decision function for the measurement data $x_{j,k}$ and θ is corrsponding threshold, which is used to determine whether an AP is normal and faulty. Thus, the detection function can be formulated as a binary classification problem as follows:

$$y_k = \begin{cases} \text{Normal} & \text{if } f(x_{j,k}) \geq \theta \\ \text{Faulty} & \text{if } f(x_{j,k}) < \theta \end{cases}, k = 1, ..., n \tag{5.4}$$

where $x_{j,k}$ is the measurement data reported by each AP, and y_k denotes the status of $ap_{j,k}$, which can be normal or faulty.

This detection problem can be solved using machine learning algorithms including classification and clustering algorithms. These algorithms are used for the faulty AP detection and will be introduced in section 5.3.

The loss function can be defined as flows:

$$L(y_k, \hat{y_k}) = \begin{cases} 0 & \text{if } y_k \neq \hat{y_k} \\ 1 & \text{if } y_k = \hat{y_k} \end{cases}, \tag{5.5}$$

5.3 Self-healing Algorithms

In this section, the machine-learning-based self-healing algorithms including K-nearest Neighbor (KNN), Local Outlier Factor (LOF), Support vector machine (SVM), K-means and Random Forest (RF) are examined and the performance is compared in terms of detecting faulty APs. The aim of the comparison is to find out which machine learning algorithm is suitable for the self-healing problem in WiFi networks. The details of these algorithms will be introduced in the following sections.

5.3.1 K-nearest Neighbor (KNN)

KNN is a supervised learning algorithm where the result of new measurement data is classified based on the majority of K nearest neighbors. Unlike the unsupervised classification approach, supervised KNN requires prior labeling of training data [87].

Let T_j denote the sequence of labeled training data during the j-th TTI, thus T_j can be written as follows:

$$T_j(n) = \{(t_{j,1}, y_1), (t_{j,2}, y_2), ..., (t_{j,k}, y_k), ..., (t_{j,n}, y_n)\}, \tag{5.6}$$

where $t_{j,k}$ is a n-tuple as defined in Equation 5.3, y_i is the label indicating if the AP is faulty or otherwise, $y_k \in \{\text{Normal, Faulty}\}$, $k = 1, 2, ...n$. Thus, T_j can be viewed as training data for the self-detection of faulty APs.

The main procedures for KNN are as follows:

(i) Calculate distance: Let $D_{knn} = (||x_{j,k} - t_{j,k}||)_{j=1}^{n}$ be the array of distance of the measured data $x_{j,k}$ to all training data $t_{j,k}$.

(ii) Find closest neighbors: Let $S_j(n)$ be the above array D_{knn} sorted in the increasing order and $T_j^{'}(n)$ denotes the corresponding training data. Let $N_K(T^{'})$ denote the first nearest K elements in $T_j^{'}(n)$.

(iii) Predict AP status label $\hat{y_k}$: Let v_i denote the label of the i-th elements in $N_K(T^{'})$. In $N_K(T^{'})$, calculate the number of elements with label v_i. The majority number of label v_i is taken as the predicted results of $x_{j,k}$.

Let $V = \{v_1, ..., v_i, ..., v_k\}$ denote the list of labels in $N_K(T^{'})$.

$I(v_i, y_k)$ is the indicator function, if $v_i = y_k, I = 1$, otherwise $I = 0$.

Let $g(v_i)$ denote the number of elements with label v_i in $N_K(T^{'})$.

$$\begin{gathered} g(v_i) = \sum_{i} I(v_i, y_k), \\ i = 1, 2, ..., K. \end{gathered} \tag{5.7}$$

Therefore, the predicted function can be described as follows:

$$\hat{y_k} = \arg\max_{v_i} g(v_i), k = 1, ..., n, \tag{5.8}$$

where $\hat{y_k}$ is the predicted AP status.

Referring back to the loss function in Equation 5.5, the aim of KNN algorithm is to reduce the error detection rate and can be described as follows:

$$\max \sum_{i=1}^{n} L(y_k, \hat{y_k}). \tag{5.9}$$

5.3.2 Local Outlier Factor (LOF)

LOF detection is an unsupervised anomaly detection algorithm that compares the local densities of sequences of incoming measured AP data. The local densities denoted by ρ of target points are calculated by LOF and used to compare with its k neighbors. The larger the difference between the sample and its neighbors is, the larger the outage factor score assigned to the sample [140].

The advantage of LOF is that no prior knowledge is required in advance to detect the unknown AP outage. The detailed definitions of LOF are explained as follows.

Let $\mathbf{x_j} = \{x_{j,1}, ..., x_{j,k}, ..., x_{j,n}\}$ denote a set of measurement data during the jth TTI, and $x_{j,k}$ is a n-tuple as defined in Equation 5.3. The goal of the unsupervised learning LOF is to predict the label $\hat{y_k}$ indicating if the AP is faulty or otherwise, $\hat{y_k} \in \{\text{Normal, Faulty}\}$, $k = 1, 2, ...n$.

The main procedures of LOF are as follows:

1. Firstly, computing the k-distance of each instance measurement $x_{j,k}$ that is denoted as a point p. k-distance is the distance of p to its k-th nearest neighbor. Let $d_k(x_{j,k})$ denote the k-distance of p. The measurement data points that lies within $d_k(x_{j,k})$ are called its k-distance neighborhood.

2. The second step is to construct a neighborhood of p by including the measurement data points that fall within the $d_k(x_{j,k})$ range. Those measurement data points are the neighbors of p, denoted as $x_{j,i} \in \mathbf{x_j}$.

3. The third step is to compute the reachability distance. The neighbors of p within $d_k(x_{j,k})$ have the same reachability distance. The reachability distance of a measurement data point $x_{j,i}$, which is outside the $d_k(x_{j,k})$ range, is the real distance between p and $x_{j,i}$. The reachability distance is used to predict a stable result of an AP's status because it can ensure the measurement data point which is far away from p has

less effect on the local reachability density. This will be explained in the next step.

Let $d(x_{j,k}, x_{j,i})$ denote the distance between p. Therefore, the reachability distance denoted by $d_r(x_{j,k}, x_{j,i})$ is equal to the maximum of $d_k(x_{j,k})$ and $d(x_{j,k}, x_{j,i})$:

$$d_r(x_{j,k}, x_{j,i}) = \max\{d_k(x_{j,k}), d(x_{j,k}, x_{j,i})\} \tag{5.10}$$

The fourth step is to compute the local reachability density of p. The local reachability density shows the distance between a measurement point and p. The measurement point is further from p when the value of local reachability density is lower.

The local reachability density ρ is the inverse of average d_r and can be defined as

$$\rho(x_{j,k}) = \frac{|N_k(x_{j,k})|}{\sum_{x_{j,i} \in N_k(x_{j,k})} d_r(x_{j,k}, x_{j,i})} \tag{5.11}$$

where $N_k(x_{j,k})$ is the number of neighbors of point p.

4. Finally, the D_{lof} represents a local density-estimation score and can be computed as follows:

$$D_{lof}(x_{j,k}) = \frac{\sum_{x_{j,i} \in N_k(x_{j,k})} \frac{\rho(x_{j,i})}{\rho(x_{j,k})}}{N_k(x_{j,k})} \tag{5.12}$$

When the value of D_{lof} is close to 1, it means the measurement data point p has same density relative to its neighbors. On the contrary, a significantly high D_{lof} score indicates the APs are faulty. The prediction function of an AP's status can be defined as follows:

$$\hat{y_k} = \begin{cases} \text{Normal} & D_{lof} < 1 \\ \text{Faulty} & \text{otherwise} \end{cases}, k = 1, .., n, \tag{5.13}$$

where $\hat{y_k}$ is the predicted AP status.

5.3.3 Support Vector Machine (SVM)

Support vector machines are a supervised learning algorithm that can be used to separate two classes of data [141] by finding an optimal hyperplane. Usually, an SVM conducts an N-dimensional hyperplane, which optimally separates the data into two categories in the feature space [142]. The vector that is the nearest to a hyperplane is called the Support Vector (SV) [143].

Let $\mathbf{x_j}$ denote the sequence of labeled training data during the j-th TTI, thus $\mathbf{x_j}$ can be written as :

$$\mathbf{x_j} = \{(x_{j,1}, y_1), (x_{j,2}, y_2), ..., (x_{j,i}, y_i), ..., (x_{j,n}, y_n)\}, \tag{5.14}$$

where $x_{j,i}$ is a n-tuple as defined in Equation 5.3, y_i is the label indicating if the AP is faulty or otherwise, $y_i \in \{\text{Normal, Faulty}\}$, $i = 1, 2, ...n$. Thus, $\mathbf{x_j}$ can be viewed as training data for the self-detection of faulty APs.

$$y_i = \begin{cases} \text{Normal} & C_1 \\ \text{Faulty} & C_2 \end{cases}. \tag{5.15}$$

where $C_m, m = \{1, 2\}$ are the classes labels, $C_m \in \{-1, 1\}$.

In SVM, the standard formulation for a two-class classification problem is as flows:

$$f(x) = \omega^T \phi(x) + b, \tag{5.16}$$

which is a linear model, where $x = \{x_{j,1}, x_{j,2}, ..., x_{j,i}, ..., x_{j,n}\}$ is the input vector and ω is the vector of coefficients for the linear model, ϕ is a general feature-space transformation function (which can eventually be nonlinear) and b represents the bias of the model.

As defined in Equation 5.14 and 5.15, each measurement vector $x_{j,i}$ is associated with one of the two classes labels (C_1, C_2) representing the AP status, C_1 is denoted as Normal AP and C_2 is denoted as Faulty AP. If $f(x^*) > 0$, the unknown measurement vector x^* belongs to class C_1. If $f(x^*) < 0$, the unknown measurement vector x^* belongs to class C_2 . The implicit assumption is that the training data is linearly separable, so that the coefficient vector ω and the parameter b can be determined (i.e., there exists at least one feasible combination of ω and b).

As the training data is assumed linearly separable, two parallel hyperplanes are used to separate the two classes of data. The larger the distance between two parallel hyperplanes is, the higher the accuracy of the predicted AP status is. Therefore, the training process of an SVM can be seen geometrically as the problem of maximizing the minimum Euclidean distance between the two parallel hyperplanes. The optimization problem can be expressed as:

$$\min \|\omega\|^2,$$

subject to :
$y_i(\omega^T \phi(x_i) + b) - 1 > 0$

(5.17)

where $y_i \in \{-1, 1\}$ is the label of AP status.

Once the model is trained, the predicted AP status for the input measurement vector x^* can be obtained by simply evaluating the sign of $f(x)$ in the original linear model $f(x) = \omega^T \phi(x) + b$, with the coefficient vector ω populated using the results from the minimization of the cost function in Equation 5.17, hence the predicted value of AP status denoted as $\hat{y}$ can be calculated as:

$$\begin{gathered} \hat{y} = f(x) = \sum_{i=1}^{n} y_i \alpha_i K(x, x_i) + b, \\ \alpha_i > 0, y_i f(x_i) - 1 > 0, \alpha_i (y_i f(x_i) - 1) = 0 \end{gathered} \tag{5.18}$$

where α_i are the Lagrange multipliers of the dual problem, and $K(x, x_i)$ is a kernel function can be Linear kenel, Polynomial kenel and Gaussian radial basis kenel (RBF).

5.3.4 Random Forest (RF)

Decision trees (DT) [144] is an algorithm to solve classification and prediction problems to use an ensemble of trees. The best way to increase the prediction accuracy of DT is to use an ensemble of trees. Random Forest is an ensemble of randomly constructed decision trees for classification and regression purposes. One of the efficient properties of the RF algorithm is that the algorithm does not tend to overfit, even if more trees are added to the forest. Breiman [145] states that the trees always converge so that overfitting is not a problem.

Let $\mathbf{x_j}$ denote the sequence of labeled training data during the j-th TTI, thus $\mathbf{x_j}$ can be written as:

$$\mathbf{x_j} = \{(x_{j,1}, y_1), (x_{j,2}, y_2), ..., (x_{j,k}, y_k), ..., (x_{j,n}, y_n)\}, \tag{5.19}$$

where $x_{j,k}$ is a n-tuple as defined in Equation 5.3 and y_k is the label indicating if the AP is faulty or otherwise. Note that for the faulty APs prediction, only binary classification is considered, so $y_k \in \{0, 1\}$ where class label 1 represents the normal AP and 0 represents the faulty AP.

A random forest's classifier consists of a collection of decision tree classifiers defined as $\{h(\mathbf{x_j}, \Theta_k), k = 1, ...\}$. Here, Θ_k represents identically distributed random vectors and each tree gives a unit vote for the most popular class at input $\mathbf{x_j}$ [145].

A decision tree in the forest is constructed by the following steps:

1. The number of trees (T) needs to be grown is chosen. Generally, the more trees are chosen, the higher the prediction accuracy of faulty AP is [146]. However, more trees need more computation time, which

is not suitable for real-time network requirements to improve handover performance. The number of trees is chosen by cross-validation based on different training dataset in self-healing. Figure 5.4c shows an example of how to choose the number of trees by cross-validation accuracy.

2. The number of features (f) splits for each node is chosen. If the feature set of the input data is denoted by F, then $f < F$ must be satisfied. The subset of features f is kept constant during the formation of the forest.

   Choosing more features increases the chance of finding a better split. However, it also increases the correlation between trees and the prediction accuracy of faulty AP will be affected by the correlation [145]. The default values are set using the square root of the total number of features for the self-healing problems [147].

T number of trees in the forest is grown with the following criteria:

1. Bootstrap samples are randomly chosen from training data and the size of the samples is n. These random selecting bootstrap samples mean the training data will be replaced by multiple times. This will reduce the entire forest variance and provide high detection accuracy.

2. To grow a tree at each node, m features are selected randomly and they are used to find the best split. The best split will ensure the faulty AP is able to be detected with high accuracy.

3. The tree is grown to the maximal extent with no pruning.

To classify the label of a sample $x_{j,k}$, a majority voting scheme is used to evaluate votes from every tree in the forest. The sample $x_{j,k}$ is passed through each tree to calculate the class for this tree. Giving a set of votes to all possible classes, the class having a maximum number of votes is considered as the predicted class of this sample.

Let $c \in \{0, 1\}$ denote the class label of one random forest tree.

Let $V = \{v_1, ..., v_i, ..., v_T\}$ denote the list of class label of the T number of trees.

$I(v_i, c)$ is the indicator function, if $v_i = c, I = 1$, otherwise $I = 0$.

Let $g(v_i)$ denote the number of trees with label v_i.

$$\begin{gathered} g(v_i) = \sum_i I(v_i, c), \\ i = 1, 2, ..., T. \end{gathered} \tag{5.20}$$

Therefore, the predicted label of $x_{j,k}$ can be written as::

$$\hat{y_k} = \arg\max_{v_i} g(v_i), k = 1, ..., n, \tag{5.21}$$

where $\hat{y_k}$ is the predicted AP status.

5.3.5 K-means

K-means is an unsupervised learning algorithm used to solve the clustering problem. The aim of K-means is to divide the measurement data into K clusters.

Let $\mathbf{x_j} = \{x_{j,1}, ..., x_{j,k}, ..., x_{j,n}\}$ denote a set of measurement data during the j-th TTI, and $x_{j,k}$ is a n-tuple as defined in Equation 5.3. The goal of the unsupervised learning K-means is to predict the label $\hat{y_k}$ indicating if the AP is faulty or otherwise, $y_k \in \{\text{Normal, Faulty}\}$, $k = 1, 2, ...n$.

The first step is to choose the number of k clusters. As the APs are needed to be detected as normal APs and faulty APs, k is chosen as two in the self-healing approach.

The next step is to randomly initialize the centroids of the clusters. The measurement data that are closest to a centroid are considered as the same cluster as described in Equation 5.22.

$$\hat{y_k} = \arg\min_{c_m} \|x_{j,k}^{(m)} - c_m\|^2 \tag{5.22}$$

where K is the number of clusters, c_m is the centroid for cluster m, $\|x_{j,k}^{(m)} - c_m\|^2$ is a chosen distance between a measurement data $x_{j,k}$ and the cluster centre c_m.

Then the centroids will be updated by calculating the average of all the measurement data in the centroid. After the new centroids are obtained, the measurement data are also updated. A loop has been generated to update the centroids. Once the centroids will not be updated. The loop of the clustering algorithm is done. Finally, the measurement data are divided into different clusters.

5.3.6 Evaluation Metrics

Some prediction metrics are used to evaluate machine learning algorithms performance. The metrics include Precision-Recall, $F1$ score and accuracy score.

Precision (P) is defined as the number of true positives (T_p) over the number of true positives plus the number of false positives (F_p). In the self-healing approach, precision can be used to measure the ratio of the number of correctly predicted normal APs to the total number of predicted normal APs. When the value of precision is high it means the number of correctly predicted normal APs is high among the predicted results. Precision (P) is defined as follows:

$$P = \frac{T_p}{T_p + F_p} \tag{5.23}$$

Recall (R) is defined as the number of true positives (T_p) over the number of true positives plus the number of false negatives (F_n). In the self-healing approach, recall is the ratio of the number of correctly predicted normal APs to the number of all normal APs in the training dataset. Recall means how many normal APs are predicted correctly among the total number of normal APs. Recall (R) is defined as follows:

$$R = \frac{T_p}{T_p + F_n} \tag{5.24}$$

F_1 Score is defined as the weighted average of precision and recall. As can be seen in Equation 5.23 and 5.24, the accuracy of AP detection cannot be measured by precision and recall when false positives (F_p) and false negatives (F_n) are the same. Therefore, F_1 is usually used to look at both precision and recall. A high F_1 score means that the prediction error is low. Thus, the high F_1 score means the accuracy of the predicted AP status is high. The F_1 score is 1 when the prediction of the AP status is all correct, while the value of F_1 is 0 when the prediction of AP status is all wrong. The F_1 score is defined as follows:

$$F1 = 2 \times \frac{R \times P}{R + P} \tag{5.25}$$

The accuracy score function computes the accuracy, either the fraction or the count of correct predictions. In multi-label classification, the function returns the subset accuracy. If the entire set of predicted labels for a sample strictly match with the true set of labels, then the subset accuracy is 1.0; otherwise, it is 0.0 [148].

The fraction of correct predictions over n_{samples} is defined as:

$$\text{Accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1_{\{\hat{y}_i = y_i\}}, \tag{5.26}$$

where $\hat{y}_i$ is the predicted value of the i-th sample, y_i is the corresponding true value and 1 is an indicator function.

Accuracy is a ratio of the correctly predicted number of observations to the total number of observations. When the dataset is not symmetric, a high accuracy alone cannot evaluate the quality of the training model. For example, 98 normal APs and 2 faulty APs are in training dataset. If the T_p is 97 and T_n is 0, the accuracy score is 97%. However, the faulty AP has not been detected at all, though the accuracy score of this model is high. Therefore, all the evaluation metrics including precision, recall, $F1$ and accuracy score have to be considered together to evaluate a machine learning algorithm in self-healing approach.

5.4 Performance Evaluation

In this section, the performance of the proposed self-healing approach using machine learning algorithms including KNN, SVM, RF, LOF and K-means is evaluated and compared with the IEEE 802.11 standard.

The self-healing approach only compared with 802.11 is because this is a new approach using self-healing to do the fault management based on NLM in WiFi networks. Self-healing is widely developed in cellular networks and hard to find a similar approach to compare in WiFi networks. However, different machine learning algorithms are used to evaluate handover performance.

5.4.1 Simulation Setup

The proposed algorithms are implemented in ns-3 with all AP operating in the 2.412 GHz band using the IEEE 802.11k protocol and each AP has four omni-directional antennas with a maximum data rate of 54 Mbps (at the physical layer). The values of simulation parameters are listed in Table 5.2. The measurement data collected includes handover delay, throughput and packet loss rate over 35 runs. Each run uses a different seed to generate a random data rate and packet size. The random traffic generation aims to satisfy different application requirements including voice call, video call, video streaming, online gaming in the different density network environment and network topologies. The averages shown are reported with a confidence interval of 95.00% under the assumption that the averages are normally distributed. This statistical methodology validates the simulation results. The simulation results are also validated by comparing proposed algorithms with conventional 802.11 standard using the same simulation parameters.

Table 5.2: Simulation Parameters

Parameter	**Value**
Simulation time (t)	100s, 120s, 180s, 300s
Speed	1 - 6m/s
Number of AP	6
Distance between two APs	20-60m
Number of Mobile Nodes	1-35
ActiveProbing	true
WiFi Standard	802.11g, 802.11k
PacketSize	10 - 10000 byte
Maximum dataRate	54 Mbps
AP transmit power	16.02dBm

Figure 5.3 shows an example of the AP placement in the simulation scenarios. There are 30 APs and 5 STAs placed in two lines and each AP is separated from its neighbors by 20-60 meters depends on different simulation scenarios. The STAs randomly move from t = 0s with different constant speeds ranging from 2m/s to 6m/s. The simulation time is 120s for each run.

5.4.2 Machine Learning Performance Evaluation

In this section, the selection and evaluation of machine learning models is explained based on one simulation scenario. In this scenario, six APs

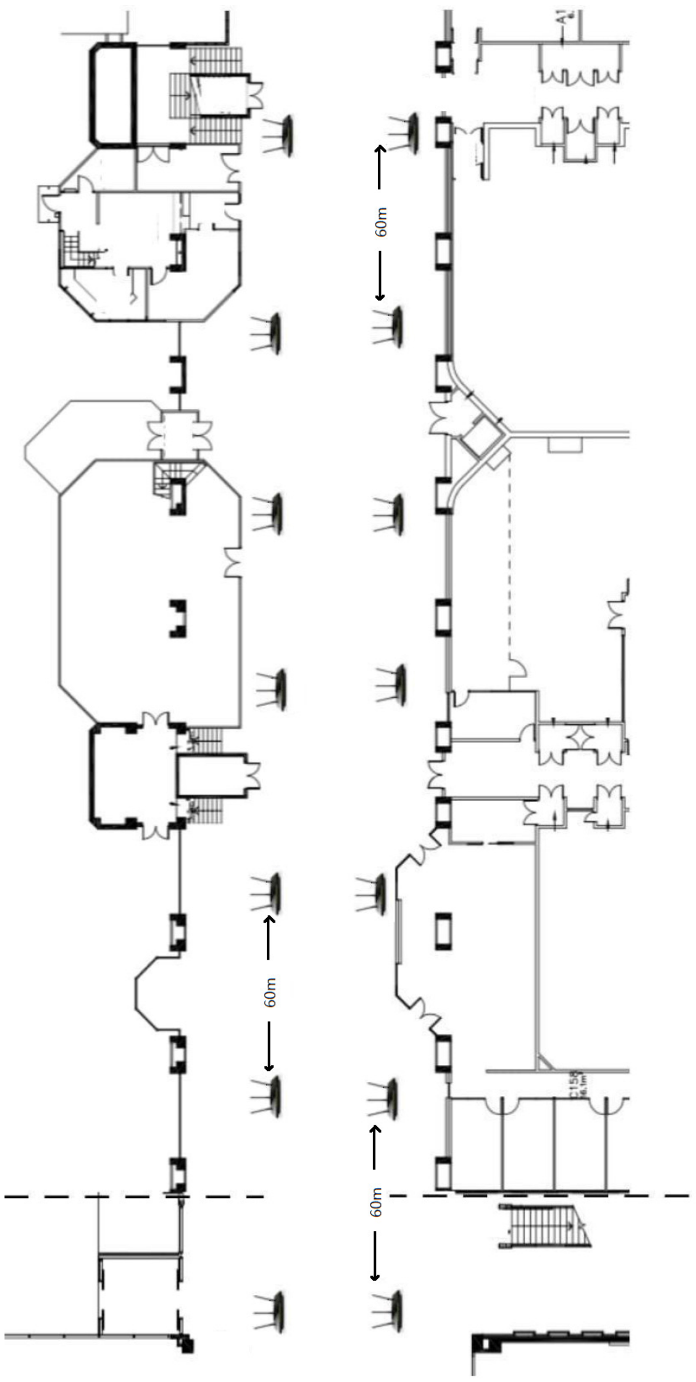


Figure 5.3: an example of AP placement in self-healing system

(AP0-AP5) are placed in one line. The distance between two APs is 60m. One STA moves from the start time of simulation t = 0s with a constant speed of 4m/s. At t = 35s, AP3 has been set to sleep status and transmit power is set to 0.1dBm to simulate hardware failures for 10 seconds. During the sleep mode, there is no energy consumed by AP3. The remaining energy logs of all APs are collected every 10 TTIs to do the temporal and spatial prediction. KNN is used to detect faulty APs. Then KNN is also compared with other classification and cluster algorithms to evaluate the network performance.

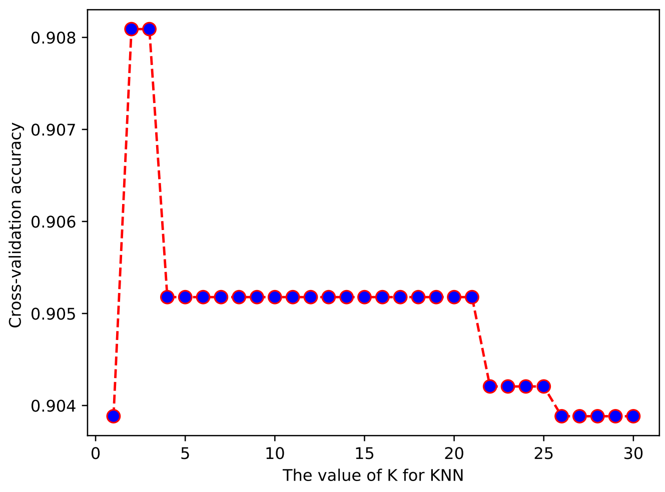
5.4.2.1 Model Selection

How the parameters of the machine learning model are evaluated and selected using cross-validation is described in section 3.2.2.1.3. In Figure 5.4a, when k= 2 or 3, the cross-validation accuracy score is highest. Thus, the optimal number of neighbors can be two or three. In Figure 5.4b, the evaluation if K clusters show that when the number of clusters is two, the cross-validation accuracy is high. The APs are divided into normal APs and faulty APs. Figure 5.4c shows the number of trees for RF. When the number of trees is two, four or six, the RF model has its high accuracy.

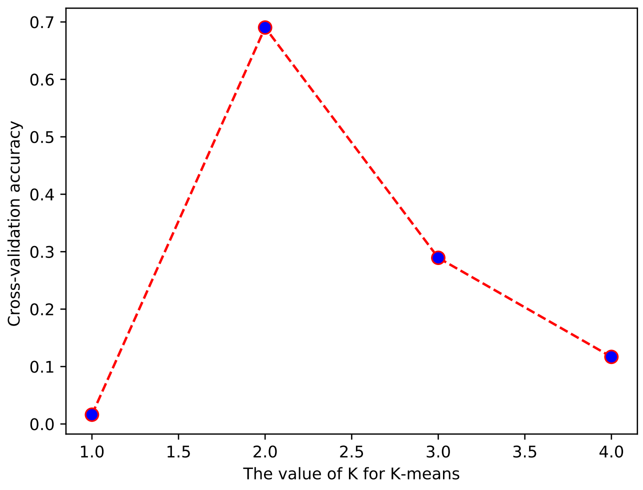
In Figure 5.5, the best value of γ is investigated using RBF kernel function. When $\gamma = 0.00012$, the SVM model shows the best performance with the highest cross-validation score.

5.4.2.2 Algorithms Comparison

In order to understand the KNN algorithm, different Classification and cluster algorithms are compared in Table 5.3. KNN, SVM and RF are supervised learning algorithms. K-mean and LOF are unsupervised learning algorithms. The results show that supervised learning algorithms have high accuracy for abnormal detection. KNN has the best performance with an accuracy of 99.28% and fast learning time is 0.01838s. The comparison

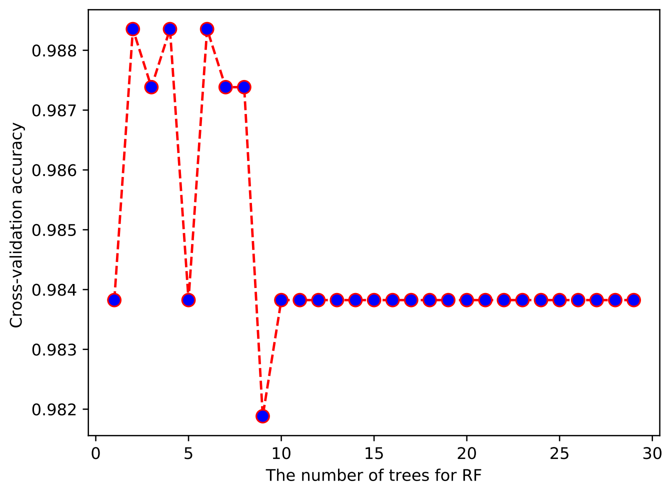


(a) Evaluating the number of neighbors K in KNN and selecting the best value of K



(b) Evaluating the number of K clusters and selecting the best value of K for K-means

results show that KNN has a high accuracy and fast learning time. This is because the KNN is a lazy supervised learning that it does not learn a training model but depends on the selected K nearest neighbors to predict



(c) Evaluating the number of trees in RF and selecting the best number of trees for RF models

Figure 5.4: Parameters evaluation and selection for KNN, K-means and RF models

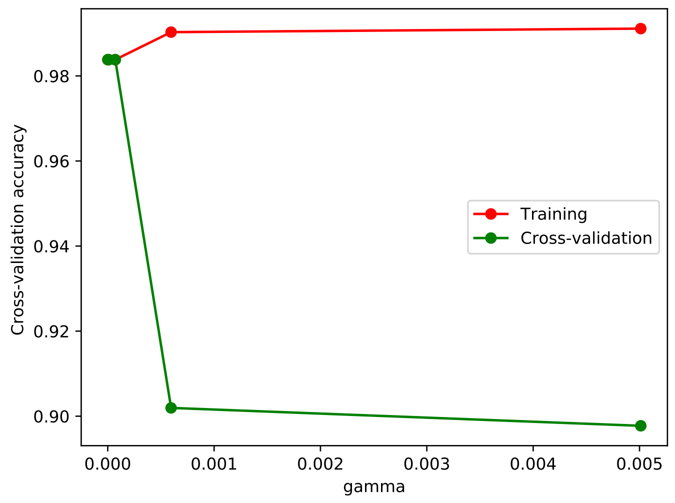


Figure 5.5: Evaluating the value of γ for SVM-kernel="rbf" and selecting the best γ to fit the SVM model

Table 5.3: Classifiers performance comparison for self-detection

Algorithms	**F-measurement**	**Accuracy**	**Learning time**
KNN	0.99	99.28%	0.01838s
SVM	0.98	98.38 %	0.10421s
Random forest	0.98	98.38%	0.03901s
K-mean	0.87	86.87 %	0.04387s
LOF	0.92	88.35%	0.02832s

the AP status. At the same time, in the self-healing system model, the feature of the dataset is a lower-dimension and KNN is more suitable for the lower-dimensional data.

5.4.2.3 Temporal Analysis

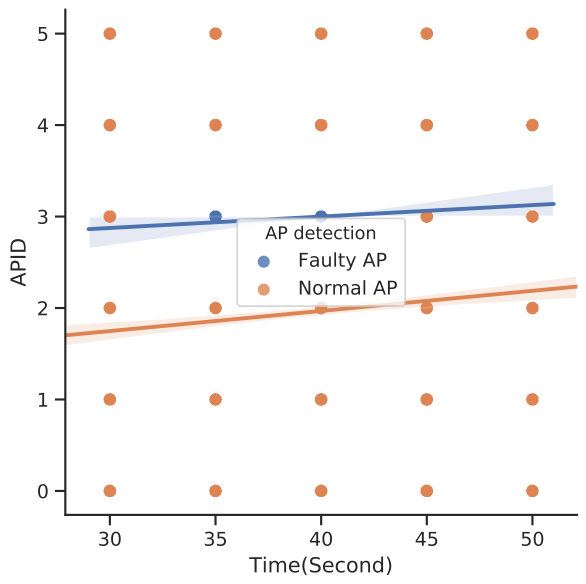


Figure 5.6: K-nearest neighbors classification is used to detect the faulty occurrence

The aim of the temporal analysis is to find out the time when a fault happened. In Figure 5.6, the x-axis shows the measurement time including the faulty time from 35s and 45s and y-axis shows the APID. The faulty AP can be detected by monitoring the remaining energy. If the remaining energy of an AP is unchanged during observation, the AP is detected as a fault because there is no energy consumed by the faulty AP. As can be seen in Figure 5.6, the blue points represent the faulty APs and the orange points are regarded as the normal APs. From the results, AP3 is detected as a faulty AP by KNN between 35s and 45s, which is in accordance with the faulty occurrence in this simulation.

5.4.2.4 Spatial Analysis

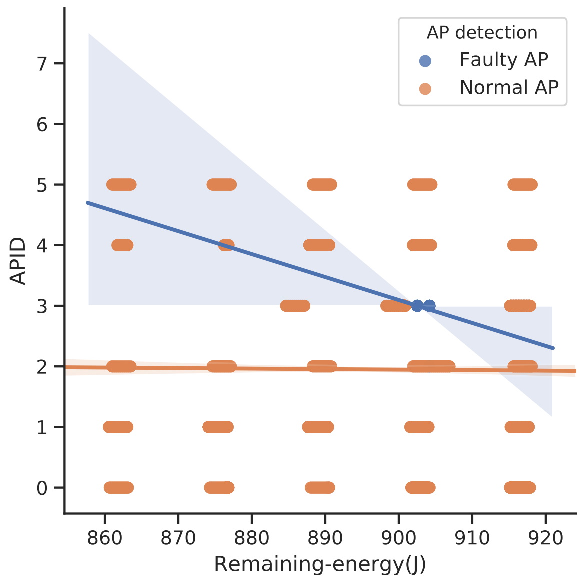


Figure 5.7: The AP status is shown at t=35s when a faulty AP is detected using KNN

The spatial analysis focuses on the temporal data of different APs at the same time period. Figure 5.7 shows the KNN result of each AP at 35s. It can be seen that the value of the remaining energy of AP3 is 904.166J during this measurement period. This means there is no energy consumed by AP3. Therefore, AP3 is in the faulty state.

5.4.3 Handover Performance Evalutation

In section 5.4.2.1, five machine learning algorithms including KNN, SVM, RF, LOF and K-means are investigated in eight network scenarios to evaluate the handover performance of the self-healing approach.

5.4.3.1 Changing the Number of Faulty APs

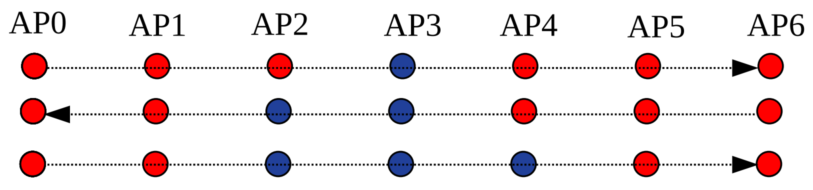


Figure 5.8: AP placement when the number of faulty APs is changed (Scenario I)

The first scenario is intended to show how the network performance is affected by the different number of faulty APs. Seven APs are placed in a line. The distance between the APs is 30m. At the simulation start time $t = 0$s, five STAs are connected to AP0. The STAs start to move between AP0 and AP6 at a constant speed of 3m/s back and forth three times within 180s.

As shown in Figure 5.8, the three lines mean that the STAs move three times. Each time, the number of faulty APs is increased by one. Therefore, AP3 is set as a faulty AP at $t = 20$s for 40s when the STAs move from AP0 to AP6 at the first time. Then, AP2 and AP3 are set as faulty APs at $t = 80$s for 40s as STAs move back from AP6 to AP0 . Finally, AP2, AP3 and AP4 are set as faulty APs at $t = 130$s for 50s when STAs move from AP0 to AP6 again. The transmit power of faulty APs is set to 0.1dBm to simulate hardware failures and the faulty APs are into sleep status.

- **Throughput**

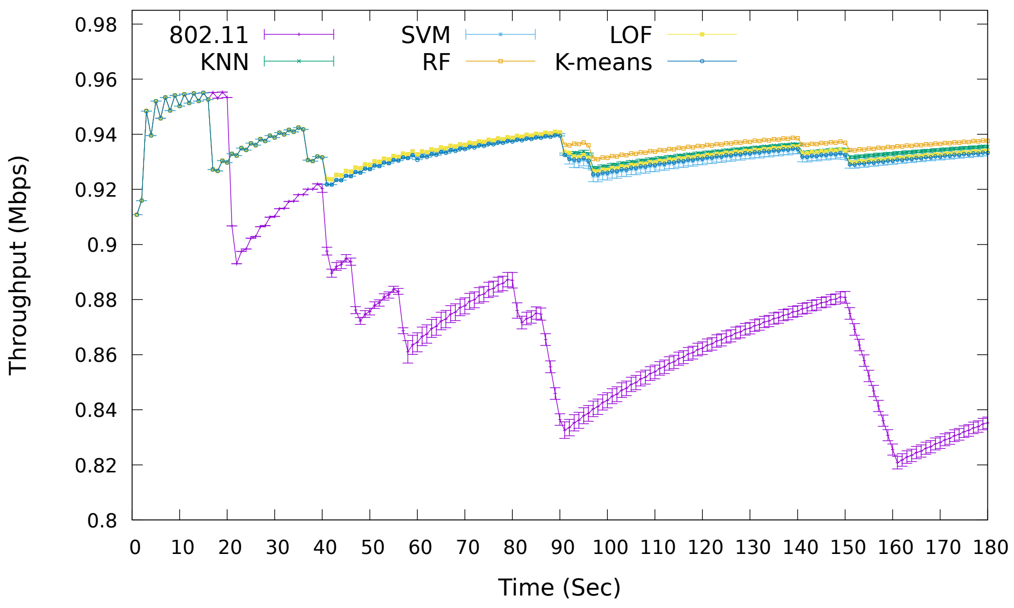


Figure 5.9: Throughput vs. Simulation time when the number of faulty APs is changed (Scenario I)

In Figure 5.9, the average throughput performance of different numbers of faulty APs is shown. At $t = 48$s, the throughput of 802.11 starts to drop because AP3 is faulty before the STAs move to AP3. Therefore, the STAs cannot associate with AP3 and this causes a longer scanning time to find a new AP to associate with. In this situation, the throughput of 802.11 standard drops by 5.07% because of one faulty AP. At $t = 90$s, the throughput of 802.11 continues to drop by 8.74% because two faulty APs cause an even a longer scanning time compared with one faulty AP. At $t = 160$s, the throughput of 802.11 dropped by 10.68% because three APs are faulty. Therefore, as the number of faulty AP increases, the average throughput performance worsens.

In order to improve the throughput performance, machine learning algorithms are used to detect faulty APs and then do the compensation for network degradation. All the compensation algorithms show an improvement over the standard 802.11. This is because the STAs are prevented from reassociating with a faulty AP and the STAs that are associated with a faulty AP are forced to handover to neighbor APs. The detection of faulty APs reduces the scanning time wasted by searching for faulty APs.

In these machine learning algorithms, Random forest (RF) shows the best performance. This is because the RF model can predict the unseen new measurement data with the highest accuracy without a problem of overfitting [145]. The training model with the overfitting problem has a higher error rate to predict the faulty AP because the training model is so dependent on that training data but cannot fit with the new measurement data. At $t = 40$s, the throughput of RF is improved by 6.82% compared with 802.11. At $t = 90$s, the throughput of RF has been improved by 12.34% compared with 802.11. At $t = 160$s, the throughput of RF has been improved by 13.61% compared with 802.11.

- **Packet loss rate**

  As can be seen in Figure 5.10, when the number of faulty APs increases, the packet loss rate of standard 802.11 increases. The packet loss rate increases for two reasons in this scenario. One reason is that the packet delivery of the STAs that are associated with the faulty AP is disturbed. The other reason is that the STAs have to scan for a long time to find a new candidate AP to associate with when the faults happen. This scanning time causes a long data disruption time. At $t = 48$s, the packet loss rate is around 8.94% if there is one faulty AP. At $t = 90$s, the packet loss rate increases to around 12.82% when the number of faulty APs increases to two. At $t = 160$s, the packet loss rate continues to increase to 13.75% compared with two faulty APs because of the three faulty APs. The more faulty APs there are, the higher the packet loss rate is.

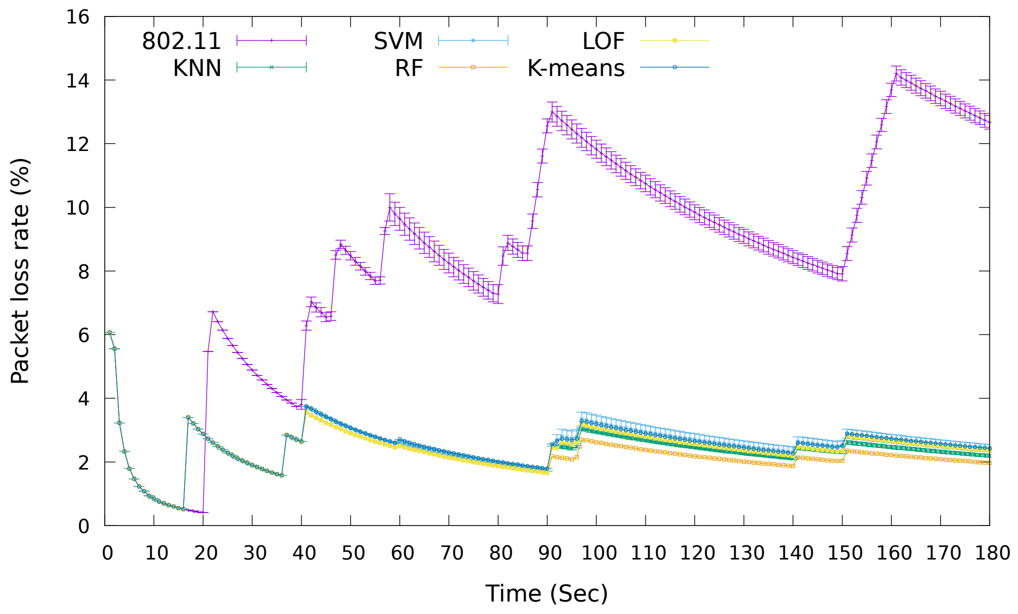


Figure 5.10: Packet loss rate vs. Simulation time when the number of faulty APs is changed (Scenario I)

With the self-healing algorithms, the overall packet loss rate has been reduced. RF shows the best improvement among these machine learning algorithms because of its fast convergence and high detection accuracy [145]. Compared with 802.11, the RF algorithm reduces the packet loss rate by 66.00% when there is one faulty AP, 87.44% when the number of faulty APs is increased to two and 85.65% when the number of faulty APs is increased to three. The packet loss rate of the self-healing approach has less affected by the number of faulty APs because the STAs are prevented from reassociating with a faulty AP.

- **Handover delay**

Figure 5.11 describes the handover delay with different self-healing algorithms. In the 802.11 standard, the total handover delay is up to 10.00s. The long handover delay is because the STAs spend a long time searching for a new AP when the number of faulty APs increases. With the introduction of self-healing algorithms, the total average handover delay of the five self-healings has been reduced by around 72.30% compared with 802.11 standard.

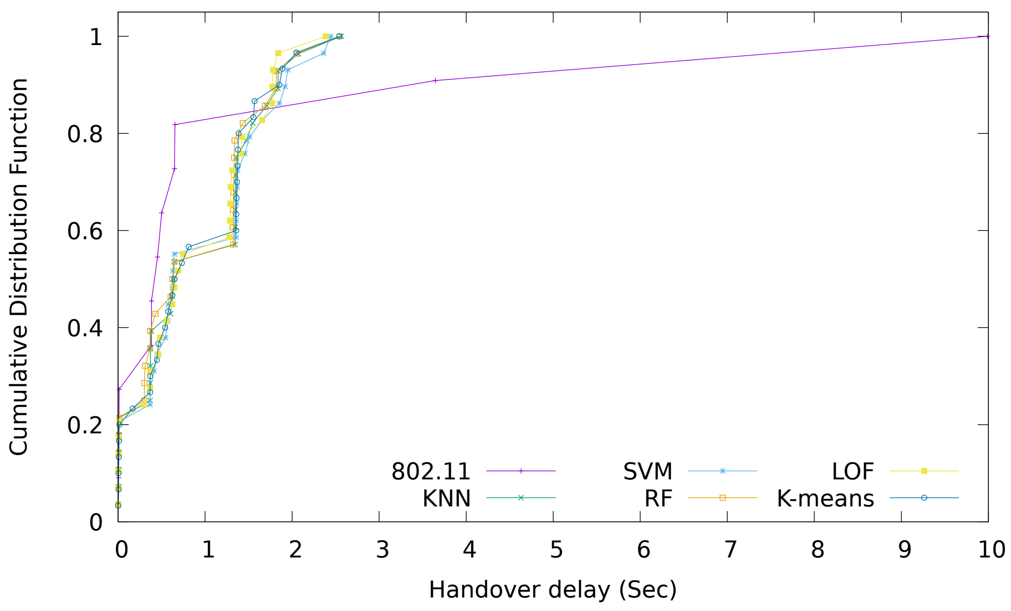


Figure 5.11: Handover delay vs. Cumulative distribution function when the number of faulty APs is changed (Scenario I)

The overall handover delay is reduced because of the following reasons. Firstly, the faulty APs has been detected and removed from the neighbor list. The STAs only scan the normal APs on the neighbor list. This reduces the scanning time wasted on faulty APs. Secondly, the scanning parameters have been updated using the approach discussed in section 4. The reduced scanning time further reduces the handover delay. Finally, the AP load information is considered in the neighbor list and this reduces the collisions between STAs to handover to neighbor APs. Therefore, the handover delay is significantly reduced by self-healing algorithms.

5.4.3.2 Changing the Positions of Faulty APs

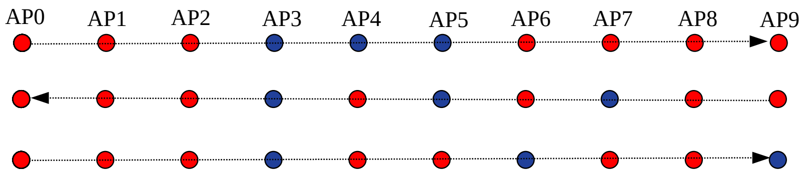


Figure 5.12: AP placement when the positions of faulty APs are changed

In the second scenario, different positions of faulty APs have been investigated by adding different hops among faulty APs. A hop is a normal AP between two faulty APs. Figure 5.12 shows the hops changing between three faulty APs, marked as blue. At $t = 0$, one STA starts to move from AP0 to AP9 at a constant speed of 3m/s back and forth three times within 300s. The three lines showed in Figure 5.12 mean the STA moves three times. The first time, the STA moves from AP0 to AP9. There is no hop between three faulty APs. The simulation time is 300s for each run. At $t = 20$s, the transmit powers of three APs (AP3, AP4 and AP5) are set to 0.1dBm to simulate hardware failures as shown in the first line. The second time, the STA moves from AP9 to AP0. There is one hop between two faulty APs. At $t = 100$s, AP3, AP5 and AP7 are set as faulty APs shown in the second line. The third time, the STA moves from AP0 to AP9. There are two hops between each pair of faulty APs. At $t = 200$s, AP3, AP6 and AP9 are set as faulty APs shown in the third line. The distance between the APs is 30m.

- **Throughput**

  In Figure 5.13, the throughput of different faulty AP positions has been investigated. In traditional 802.11 networks, at $t = 54$s, the STA loses connection because there are three faulty APs. The average throughput of 802.11 dropped by 27.08%. At $t = 140$s, the throughput of 802.11 dropped by 23.80% when there is one hop between two

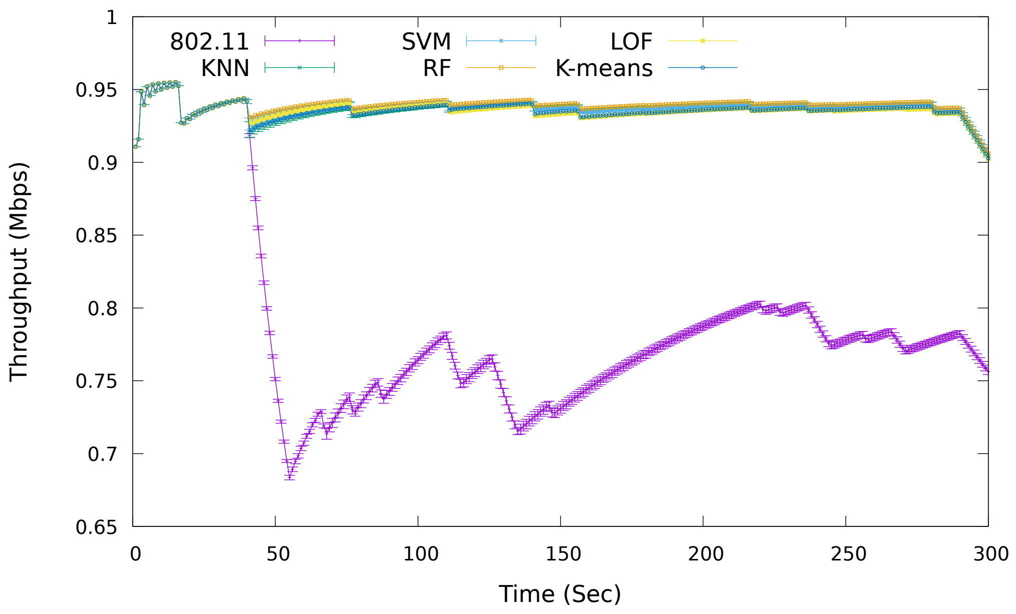


Figure 5.13: Throughput vs. Simulation time when the positions of faulty APs are changed (Scenario II)

faulty APs. At $t = 270$s, the throughput of 802.11 dropped by 18.21% when there are two hops between faulty APs. When the number of hops increases, the performance of average throughput increases. This is because the distance is reduced between the normal APs. The STA can search for APs with better throughput service to associate with. However, the long scanning time still degrades throughput performance.

The average throughput has been improved significantly by the self-healing algorithms. The throughput has not been affected a lot by the hops changing between faulty APs. The reason is that the STA has been advised the faulty APs information by APC in advance. The scanning time is reduced by removing the faulty APs from the neighbor list. Therefore, the STA is prevented from associating with a faulty AP. The supervised learning algorithm RF shows the best performance among these algorithms. With the fast detection and high accuracy rate of RF, the faulty AP can be found as soon as the outage happens. When the faulty AP is found, then STA reassociates to neighbor APs to compensate for the network degradation. As can be seen in Figure 5.13, at $t = 54$s, the throughput of RF has been improved by 34.98% compared with 802.11. At $t = 140$s, the throughput of RF has been improved by 23.26% compared with 802.11. At $t = 270$s, the throughput of RF has been improved by 22.13% compared with 802.11. Compared with RF, other algorithms also shows improvement in throughput compared to 802.11. As unsupervised learning algorithms, LOF and K-means have a lower accuracy rate compared with supervised learning. Therefore, the improvement using these two algorithms are less than supervised learning KNN, SVM and RF.

- **Packet loss rate**

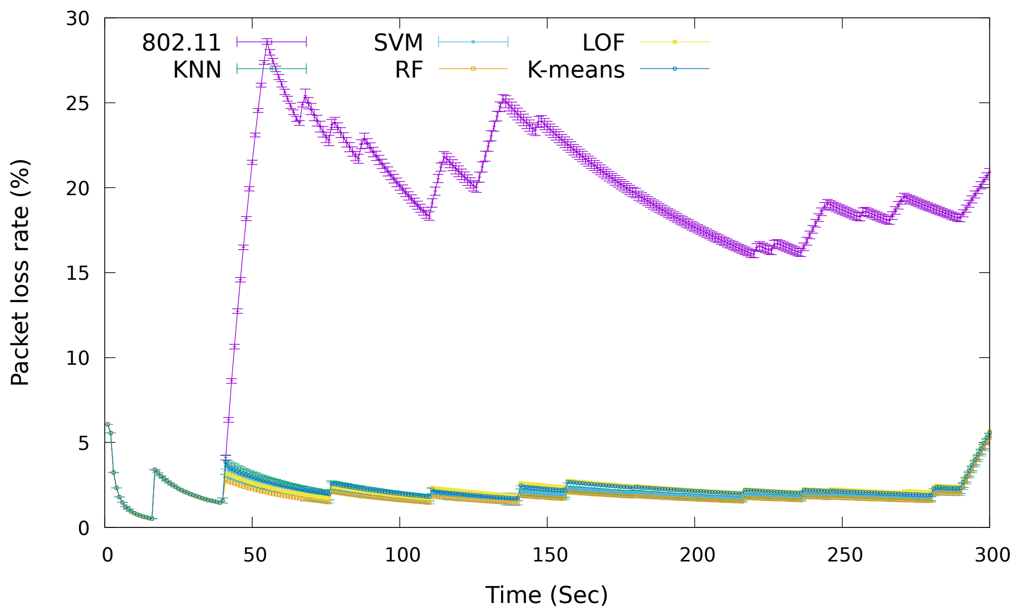


Figure 5.14: Packet loss rate vs. Simulation time when the positions of faulty APs are changed (Scenario II)

When the number of hops between two faulty AP increases, the packet loss rate becomes lower. At $t = 54$s, the packet loss rate is around 28.49% when there is no hop between faulty APs. At $t = 140$s, the packet loss rate drops to 24.44% when there is one hop between faulty APs. At $t = 270$s, when the number of hops between faulty APs is two, the packet loss is 19.37%. As the number of hops is increased, it is easy for STA to find the best neighbor APs. This is because of reducing the distance between normal APs. The data disruption time is reduced by reducing the density of faulty APs.

Compared with standard 802.11, the overall packet loss rate is reduced. RF achieves the best results among the five machine learning algorithms because the RF algorithm depends on the diversity of trees in the forest. This provides a high level of learning accuracy because of the diversity of features and RF does not tend to overfit even when more trees are added, as discussed in section 5.3.4. Without the problem of overfitting, the RF algorithm can predict the faulty AP fast with less predict errors.

Compared with 802.11, the RF algorithm reduces the packet loss rate by 93.39% when there is no hop between faulty APs. The packet loss rate is reduced by 94.56% when there is one hop between faulty APs. The packet loss rate is reduced by 92.13% when the number of hops between faulty APs is two. Overall, the average packet loss rate of self-healing algorithms is less affected when the number of hops between faulty APs changes. This is because the STA performs handover to normal APs once it obtains the information of faulty APs sent by the APC. The data disruption time is reduced by the seamless handover. Thus, the packet loss rate is reduced by shortening the data disruption time caused by faulty APs.

- **Handover delay**

  Figure 5.15 shows the handover delay with different self-healing al-

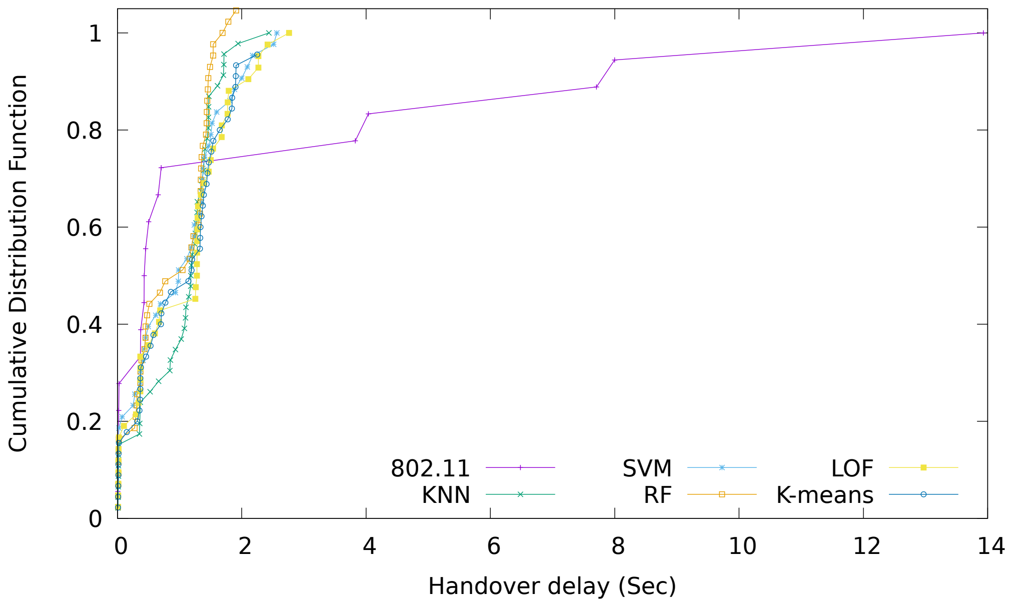


Figure 5.15: Handover delay vs. Cumulative distribution function when the positions of faulty APs are changed (Scenario II)

gorithms. In the 802.11 standard, the total handover delay is increased to 14.00s, caused by the faulty APs. The high handover delay is because the STA takes a long time to find a new AP to associate with when the outage happens. The long scanning time searching for the target AP generates a long handover delay. When there is no hop between three faulty APs, it is hard for the STA to find the target AP than when there is one hop between faulty APs because the distance between normal APs is too large and the STA cannot search for available neighbor APs. The total handover delay of self-healing has been reduced by around 80.32% compared with standard 802.11. This is because the STA can know the information of faulty APs in advance and are prevented from scanning the faulty APs. Thus, handover delay and failure handover rate are reduced. Also, the STA associated with the faulty AP can handover to neighbor APs to compensate for network degradation.

5.4.3.3 Changing the Transmit Power of APs

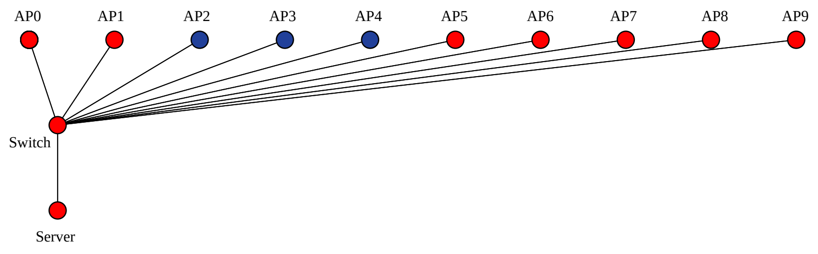


Figure 5.16: AP placement when the transmit power of APs is changed (Scenario III)

In the third scenario, there are two different values of transmit power is used to evaluate the self-healing algorithms. In Figure 5.16, 10 APs are placed in a line. The distance between APs is 30m. The simulation time is 100s for each run. At $t = 0$s, one STA moves with a constant speed of 3m/s from AP0 to AP9. The transmit power is set to 6.0206dBm and the coverage range of each AP is 30m. At $t = 10$s, AP2, AP3 and AP4 (blue nodes) are set to sleep status and transmit power is set to 0.1dBm to simulate hardware failures for 90 seconds. The STA loses connection when STA moves from AP2 to AP4. This is because the distance between AP2 and AP4 is 90m that is out of the coverage range of 30m. In this situation, the self-healing cannot do compensation for the network degradation caused by faulty APs. Therefore, the transmit power has to be increased to 16.02dBm to compensate for network degradation with self-healing algorithms together.

- **Throughput**

  Figure 5.17 shows the average throughput performance when the transmit power is changed. When the transmit power is set to 6.02dBm, the self-healing algorithms cannot improve throughput performance. This is because the STA wants to handover to neighbor APs when it

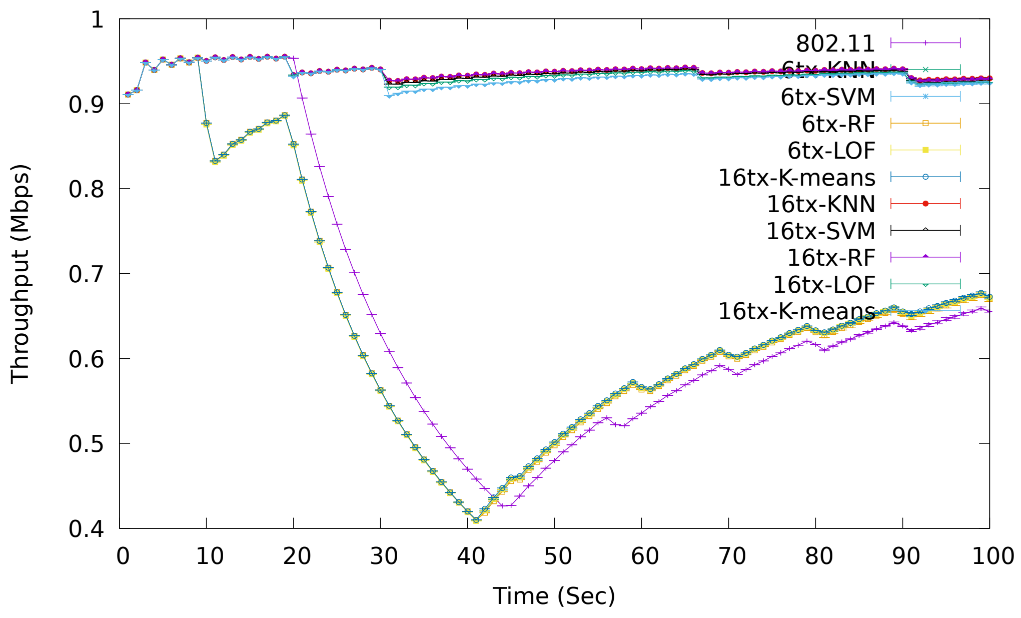


Figure 5.17: Throughput vs. Simulation time when the transmit power of APs is changed (Scenario III)

moves to AP2. However, AP2, AP3 and AP4 are set to faulty APs before STA reaches AP2. The AP coverage range is 30m but the distance between AP2 and AP5 90m.

Although the self-healing algorithms can detect the three faulty APs, the best candidate AP in the neighbor list cannot be obtained because it is out of coverage range. Thus, the STA throughput performance is still suddenly degraded when it moves to AP2. In this situation, the transmit power has to be increased to make sure the STA is within the coverage of all the neighbor APs. This can allow the STA to perform handover successfully and keep connectivity. After the transmit power is increased to 16.02dBm, the STA can handover to AP5 even though AP2, AP3 and AP4 are faulty APs. The overall average throughput has been improved by 42.25% compared with standard 802.11 by using self-healing algorithms.

- **Packet loss rate**

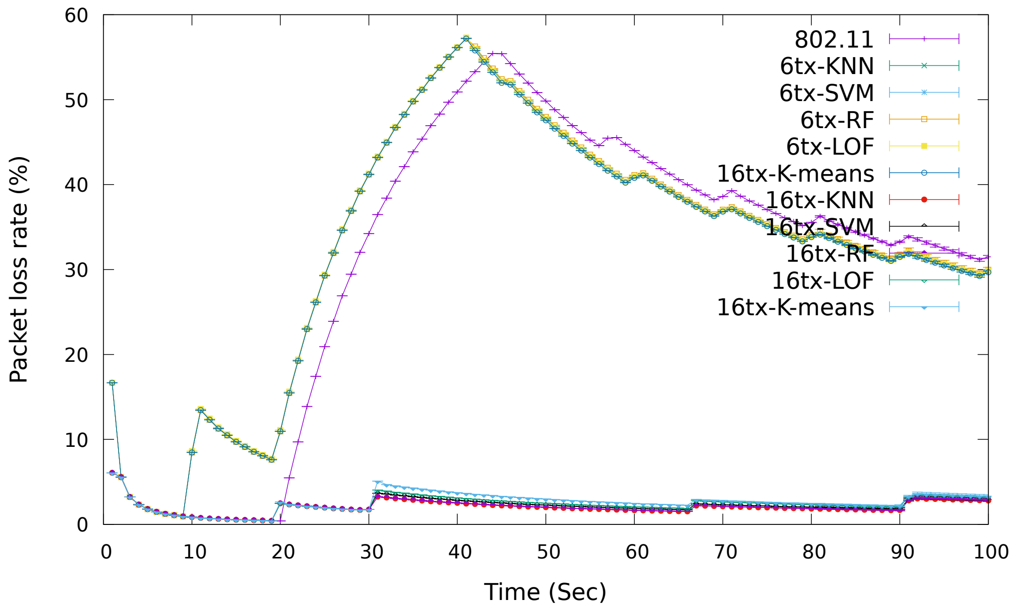


Figure 5.18: Packet loss rate vs. Simulation time when the transmit power of APs is changed (Scenario III)

As can be seen in Figure 5.18, the packet loss rate cannot be improved by self-healing algorithms when the transmit power of each AP is 6.0206dBm. The data connection is disturbed because of handover failure. The reason causes handover failure is the STA cannot find a target AP to associate with. As the coverage range of each AP is 30m, the STA is out of coverage range when it moves from AP2 and AP4. Thus, the STA loses connectivity. Although the self-healing algorithms can detect the faulty AP in advance, the STA still cannot handover to neighbor AP. This is because the STA is out of the range of the nearest normal AP that is AP5. Thus, the data connection is disrupted and causes a high packet loss rate. After increasing the transmit power to 16.02dBm, almost all the self-healing algorithms achieve the same improvement and reduce the average packet loss rate by round 93.42%.

- **Handover delay**

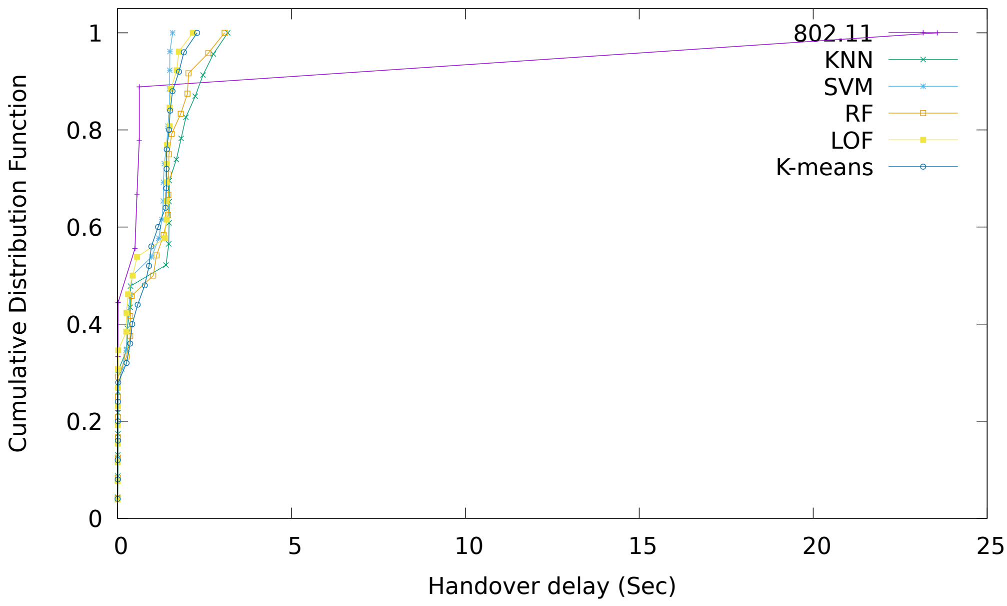


Figure 5.19: Handover delay vs. Cumulative distribution function when the transmit power of APs is changed (Scenario III)

The total handover delay of the 802.11 standard is up to 24.28s. As AP2, AP3 and AP4 are set as faulty APs before the STA moves to AP2, it spends long scanning time to find a new AP to associate with. The long scanning time causes handover failure. Thus, the handover delay is long in traditional 802.11 networks. In self-healing algorithms, when the faulty APs are detected, the neighbor list is updated. However, there is no AP on the neighbor list that can be associated with by the STA before it moves to AP5 because the coverage area is 30m when the transmit power of AP is 6.0206dBm.

In this situation, the handover delay cannot be reduced by only using self-healing algorithms. Therefore, the transmit power is increased to increase the coverage range of each AP. As can be seen in Figure 5.19, compared with standard 802.11, the total handover delay of self-healing is reduced to around 83.33% when the transmit power increases to 16.02dBm. With the increased transmit power, the STA can find a new target AP to associate with and the total handover delay is reduced.

5.4.3.4 Changing the Faulty Setting Time for Faulty APs

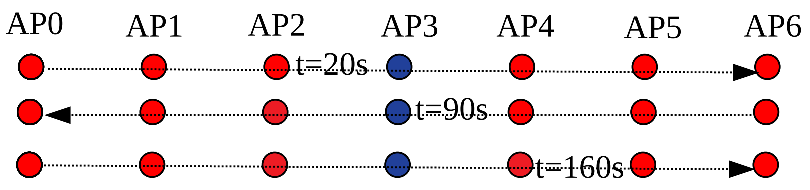


Figure 5.20: AP placement when the faulty setting time is changed (Scenario IV)

In the fourth scenario, the time of setting an AP to fault is investigated. Seven APs are placed in a line. The distance between APs is 30m. The transmit power of each AP is 16.02dBm. When AP3 is set as faulty AP, it is in sleep status and the transmit power is set to 0.10dBm to simulate hardware failures. During the sleep mode, there is no energy consumed by AP3. The Figure 5.20 shows the AP placement of this scenario. The blue node, AP3, is the faulty AP.

There are three different times to set faulty AP when the STA is moving. Figure 5.20 shows when AP3 is set as faulty AP. One STA moves from AP0 to AP6 back and forth three times within 180s. As shown in Figure 5.20, the three lines mean the STA moves three times. At the start of the simulation $t = 0$s, the STA is connected to AP0 and the STA starts to move from AP0 to AP6. Firstly, at $t = 20$s, when the STA moves to AP2, AP3 is set as faulty AP. Then, when the STA moves back from AP6 to AP0, at $t = 90$s, AP3 is set as faulty just at the same time STA moves to AP3. Finally, the STA moves from AP0 to AP6 again, and at $t = 160$s the STA moves to AP4 but at this time AP3 is set as faulty AP. The simulation time is 180s for each run.

- **Throughput**

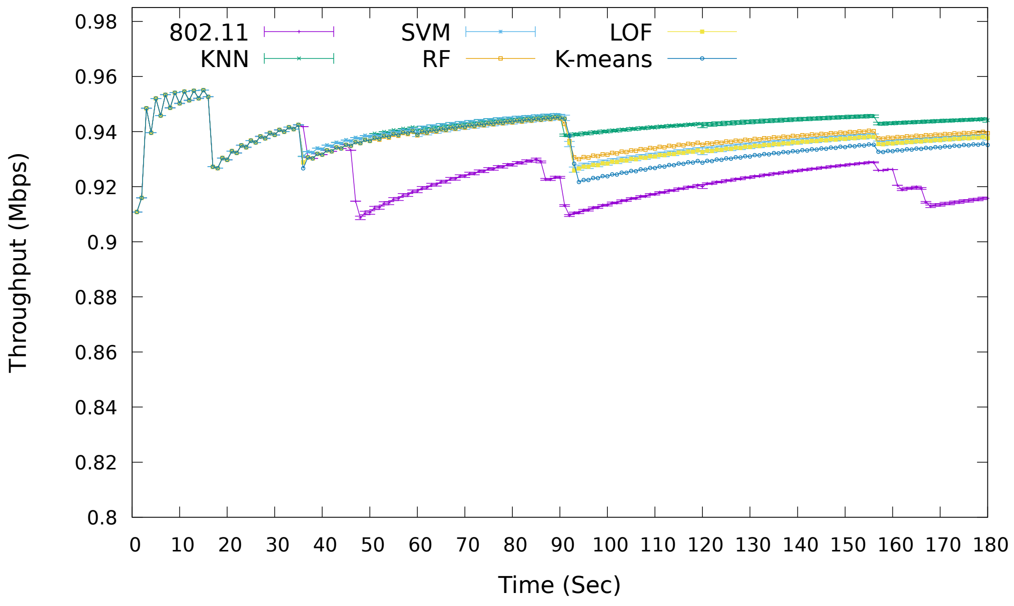


Figure 5.21: Throughput vs. Simulation time when the faulty setting time is changed (Scenario IV)

Figure 5.21 shows the throughput comparison of different faulty time of AP3. In 802.11, at $t = 48$s, the throughput drops by 0.39% because AP3 is faulty before the STA moves to it, and at $t = 92$s, the throughput drops by 0.38% because AP3 is set as a faulty AP at the same time as the STA moves to AP3. At $t = 170$s, the throughput of 802.11 drops by 0.39% when AP3 is set to fault the third time. The throughput performance of these three different faulty times of AP3 is almost the same in this scenario.

With the self-healing approach, the average throughput has been improved by all the machine learning algorithms. KNN shows the best performance in this scenario. This is because KNN is a lazy supervised learning algorithm. The lazy supervised training dataset is the training model, so it learns very fast [149]. Therefore, KNN is more suitable for this scenario with one AP becoming faulty at different time. Compared with standard 802.11 , the average throughput has been improved by 1.80% with KNN algorithm.

- **Packet loss rate**

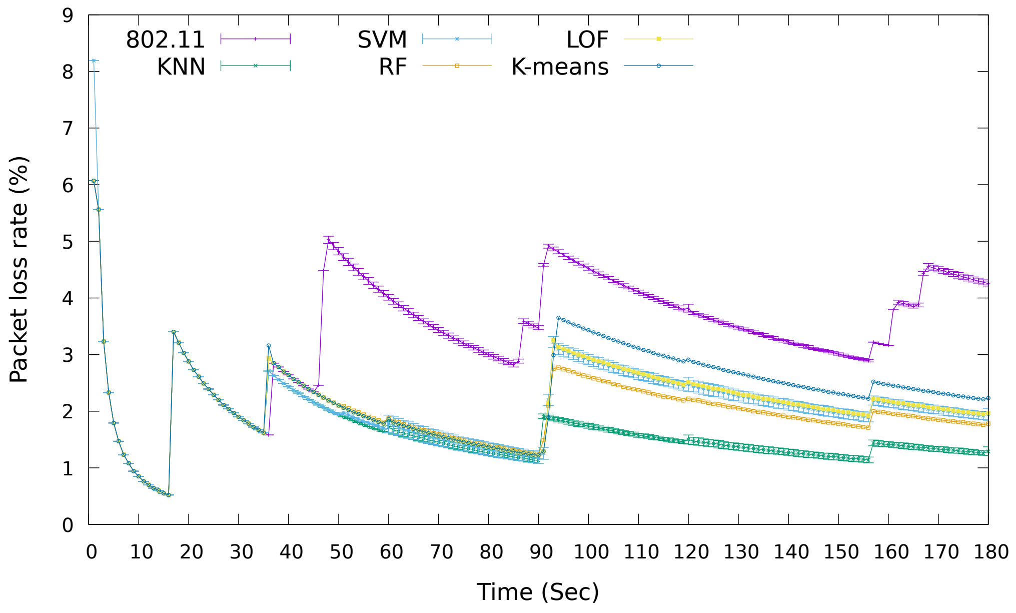


Figure 5.22: Packet loss rate vs. Simulation time when the faulty setting time is changed (Scenario IV)

Figure 5.22 shows how the packet loss rate is affected by different faulty time of AP3. In standard 802.11, the packet loss rate increases almost the same at different faulty time. Each time, the packet loss rate is increased by around 8.02% when the outage happens. At $t = 48$s, the packet loss rate increases because AP3 is set as a faulty AP before the STA moves to it. The faulty AP3 causes the STA to spend a long time searching for a target AP to associate with. At $t = 92$s, AP3 is set as faulty AP at the same time when the STA moves to AP3. In this case, AP3 is considered as the target AP for the STA to associate with. The fault causes the STA to be unable to perform a handover to AP3 and to scan for a new target AP. The additional scanning time generates the disruption of data transfer. At $t = 170$s, the increased packet loss rate is because the STA is de-associated from AP3 and starts to scan for a new AP to associate with. The suddenly disturbed data connection generates a high packet loss rate.

Compared with standard 802.11, the average packet loss rate is reduced by 51.63% with KNN, by 37.98% with SVM, by 37.09% with LOF, by 40.65% with RF and by 32.04% with K-means. The supervised learning algorithms, KNN and RF show better performance than other algorithms because of higher detection efficiency. In different faulty time, the faulty AP needs to be detected quickly to avoid losing the data connection. With KNN it is easier to detect the single AP status using its k neighbors without spending the time to train the training models. Thus, KNN is still superior to RF. The unsupervised algorithm K-means shows the worst performance. This is because one major drawback of K-means is it often falls in local optima [150]. Therefore, the faulty AP cannot be detected correctly. This can cause the packet loss rate to increase if the detection is not correct.

- **Handover delay**

  Figure 5.23 shows the comparison of handover delay at different faulty time of AP3. In this scenario, the total handover delay of 802.11 is smallest. Although there is one faulty AP causing throughput to drop and the packet loss rate to increase, the STA still maintains data connection during the simulation time. In self-healing, when the faulty AP is detected, the STA is forced to handover to neighbor APs. The number of handover increased is to improve throughput and reduce packet loss. However, increasing the number of handover leads to total handover delay being increased.

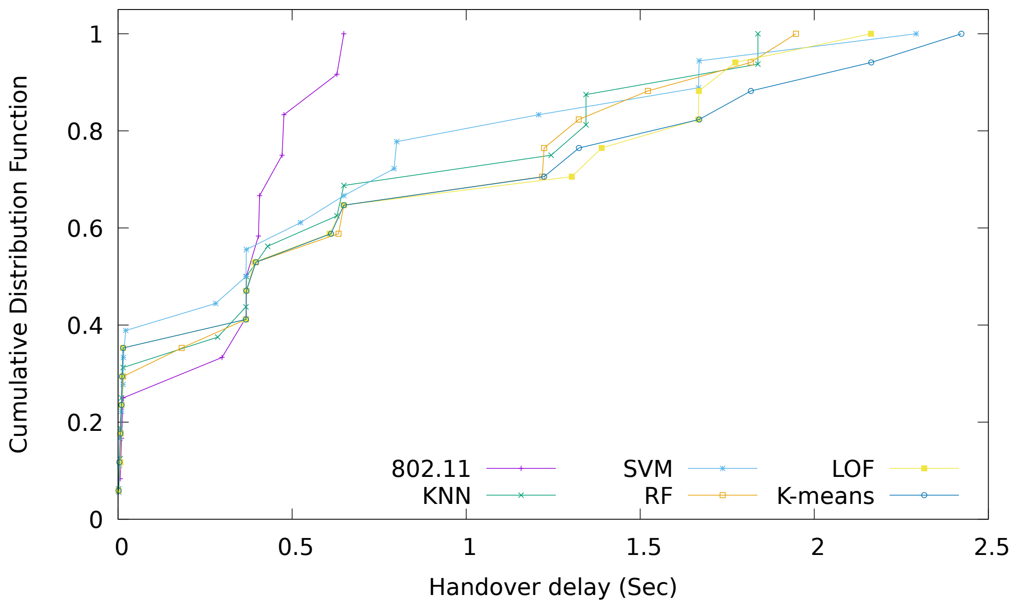


Figure 5.23: Handover delay vs. Cumulative distribution function when the faulty setting time is changed (Scenario IV)

The handover delay of KNN is the smallest among all the five machine learning algorithms. Compared with SVM and RF algorithms, the faulty AP is easily detected with KNN when the new measurement data is collected without changing the parameters of the training models. The calculation involved with updating training model behavior is largely reduced. Thus, the handover delay is reduced by fast detection.

5.4.3.5 Changing Speed

In the fifth scenario (Scenario V), how the handover performance changes with increasing STAs speed is investigated. The speed of STAs takes a value between 2m/s and 6m/s within the distance of 360m. At the simulation start time $t = 0$, the STAs are on the same location and are connected to the same AP number 0. The scenario is set up with six APs and five STAs. The distance between each AP is 60m. The simulation time is 60s for each run. The STAs start to move from AP0 to AP5 at t=0s with a constant speed between 2m/s and 6m/s. At the beginning ($t = 0$), STAs are associated with AP0. At t = 35s, AP3 is set to sleep status and the transmit power is set to 0.1dBm to simulate hardware failures for 25 seconds. During the sleep mode, there is no energy consumed by AP3.

- **Throughput**

  In Figure 5.24, the throughput is evaluated by comparing different machine learning algorithms with standard 802.11. There is no self-healing mechanism in standard 802.11. The STAs associated with faulty a AP are de-associated when an outage happens in WiFi networks. It is hard for the STAs to scan and re-associate with new APs. Therefore, the average throughput decreases by 37.42%.

  With the self-healing approach, all the machine learning algorithms improve throughput. However, the average throughput of the proposed KNN algorithm shows the best performance. The average throughput has been improved by 63.64% compared with standard 802.11. This is because the KNN can detect the faulty AP faster than other algorithms with high accuracy. K-means shows the worst performance detecting the faulty AP because when the speed of the STAs increases, the cluster approach cannot detect the fault quickly with high accuracy.

- **Packet loss rate**

  When the AP is faulty, it usually causes a high packet loss rate due to the momentary loss in connectivity. In the results shown in Figure 5.25, the packet loss rate of machine learning algorithms including KNN, SVM, RF, LOF and K-means are compared with the standard 802.11 as the STAs are moving at the speed 2, 3, 4, 5 and 6m/s.

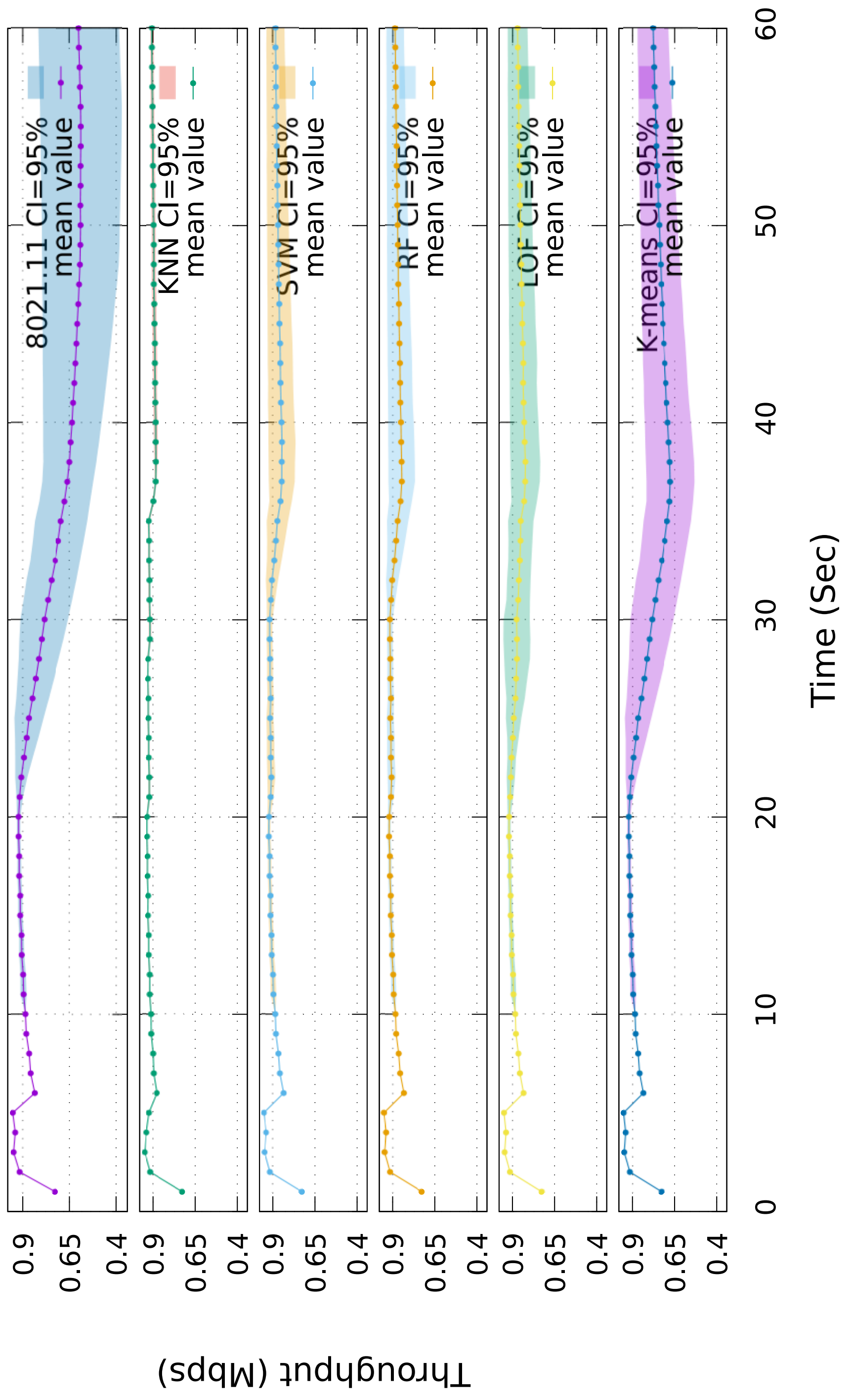


Figure 5.24: Throughput vs. Simulation time when the speed of STAs is changed (Scenario V)

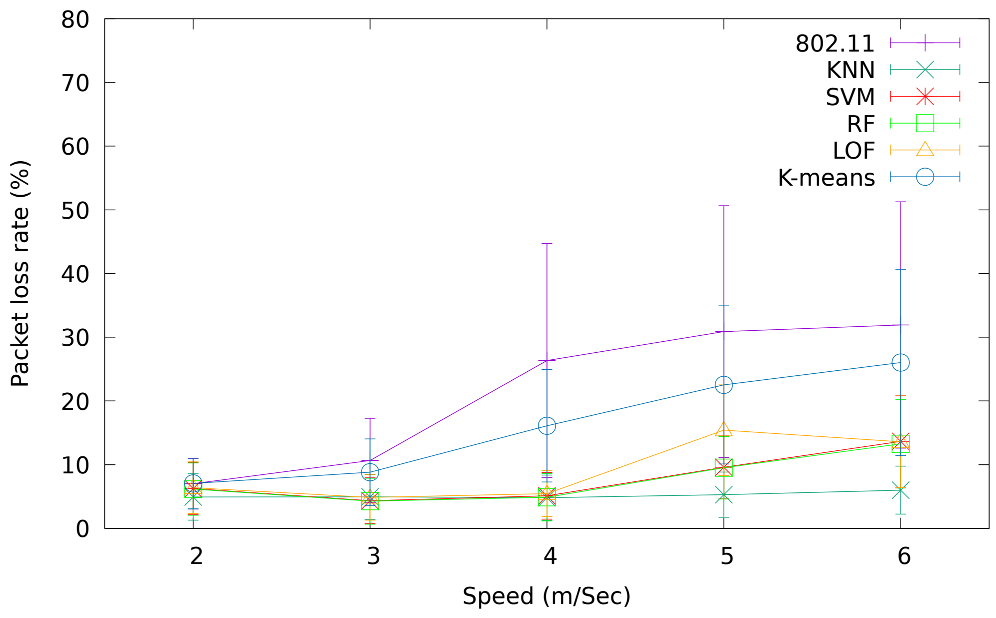


Figure 5.25: Packet loss rate vs. Speed when the speed of STAs is changed (Scenario V)

The traditional 802.11 standard shows very high average packet loss rate around 22.18%. When the speed of the STAs is 2m/s, the packet loss rate is not affected even though AP3 is a faulty because the STAs move before AP3 during the faulty occurrence. When the moving speed of STAs is 3, 4, 5 and 6m/s, the STAs move close to AP3 and try to re-associate with it. The fault of AP3 cause the STAs cannot associate with AP3 and generates high packet loss.

With the self-healing algorithms, the packet loss rate of KNN is 16.62% lower than standard 802.11. The results of KNN are not affected by varying STA speed. This is because KNN is more suitable for dynamic environments that require frequent updates of the training data [151, 152]. KNN is also an efficient learning algorithm and has successfully been used in real-applications [153]. With these advantages, the faulty AP can be detected efficiently by KNN to reduce the data disruption time. When the speed of STAs is very slow, the

SVM, random forest and LOF show performance as good as KNN. However, when the speed of the STAs increases, the accuracy of the other three supervised learning algorithms decreased. This is because SVM and RF need more learning time than KNN to update the training models. Therefore, KNN is more suitable for self-detection in the changing speed scenario.

- **Handover delay**

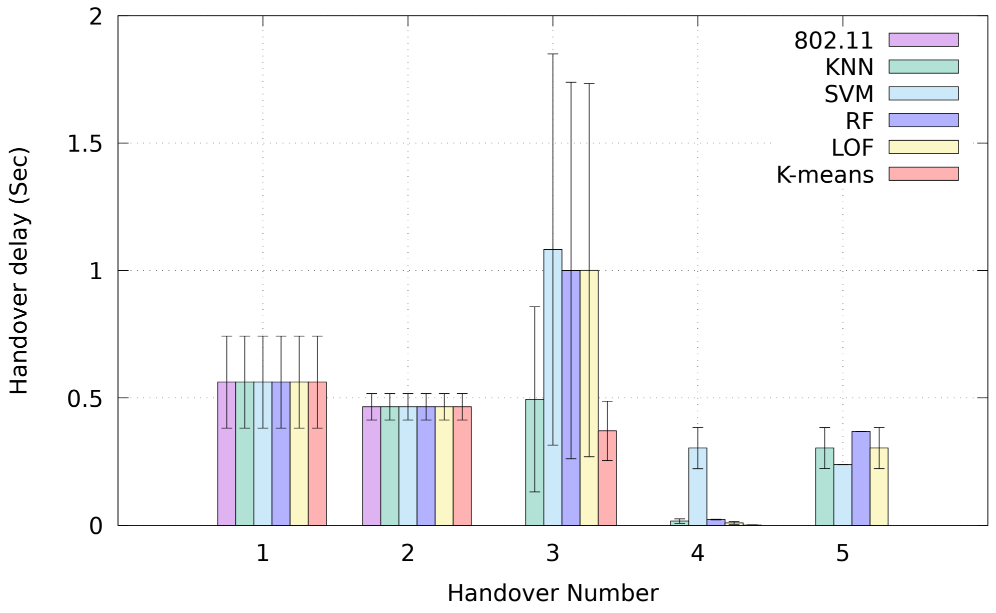


Figure 5.26: Handover delay vs. Handover number when the speed of STAs is changed (Scenario V)

Figure 5.26 shows the handover delay vs. the average number of handover is recorded. When the number of handovers is low (one or two handovers), all six mechanisms achieve almost identical handover delays, which is because there is no outage AP in the two handovers. As the STAs continue to move to AP3, the outage happens. When the speed of the STAs is 2m/s or 3m/s, the STAs only performs two handovers and the outage does not have any effect on the handover numbers. When the moving speed of STAs is 3, 4, 5 and 6m/s, the STAs move close to AP3 and try to re-associate with it at the third handover. However, the handover fails because the STAs lost connectivity of 802.11. Therefore, there is no reading for 802.11 since the third handover.

The handover is restored by self-healing algorithms at the third handover. At the fourth handover, the average handover delay of self-healing is lower compared with the fifth handover because the STAs do not perform handover when the STAs move at speed of 4 and 5m/s. Although K-means detects the faulty AP with the lowest handover delay at the third and fourth handovers, the STAs lose connectivity (hence no reading) at the fifth handover because of a lower detection accuracy rate when the speed increased to 5m/s or 6m/s.

Overall, KNN achieves the lowest average handover delay among the five machine learning algorithms. The advantage of KNN is that no assumption of a global training model is required but only a metric on the data domain [154]. Therefore, KNN can detect the faulty AP more quickly than SVM and RF. Compared with LOF, KNN still shows better performance though they are both based on computing the distance between neighbors. LOF is not able to detect the faulty AP with higher accuracy than KNN due to the low-density of APs in this senarios [155]. Once the STAs obtain the faulty AP information sent by the APC, the STAs can be prevented from associating with faulty APs. At the same time, the scanning delay can be reduced by the updated neighbor list removing the faulty APs. The STA will re-associate with the best candidate on the neighbor list.

5.4.3.6 Changing the Number of APs

In Scenario VI, the density of the network has been investigated by changing the number of APs. One STA moves between a different number of

APs(4, 6, 8, 10, 12) at a constant speed of 3m/s within the distance of 180m. At $t = 25$s, AP3 has been set to sleep status and transmit power is set to 0.1dBm to simulate hardware failures for 40 seconds. During the sleep mode, there is no energy consumed by AP3. The simulation time is 60s for each run.

- **Throughput**

  In Figure 5.27, the throughput of five machine learning algorithms including KNN, SVM, RF, LOF and K-means has been compared with standard 802.11. With the self-healing approach, all the machine learning algorithms show an improvement of throughput. However, the average throughput of proposed KNN, SVM, and RF methods are improved by 4.69%, 4.54% and 4.59% respectively compared with the IEEE 802.11 standard. The average throughput of LOF and K-means only improved by 4.24% and 4.11% respectively.

  K-means is a clustering algorithm that divides the AP into different clusters. However, K-means is not stable in a dynamic network environment when the number of AP is changed because it is not robust to outliers, especially with only one faulty AP in this scenario [156]. Another drawback of K-means is the local minima problem, where it is possible to reach a local minimum but it may not be a global optimum [157]. These drawbacks of K-means lead to the STA not being to obtain the faulty AP information in advance and causing handover failure with faulty APs. Thus, the performance of throughput degrades.

  Although LOF is a local density-based algorithm, it can reduce the local minima problem of K-means. LOF may miss some potential outliers whose local neighborhood density is very close to that of its neighbors [158]. This reduced detection accuracy means LOF cannot improve throughput performance significantly.

  Figure 5.28 shows the throughput affected by increasing the num-

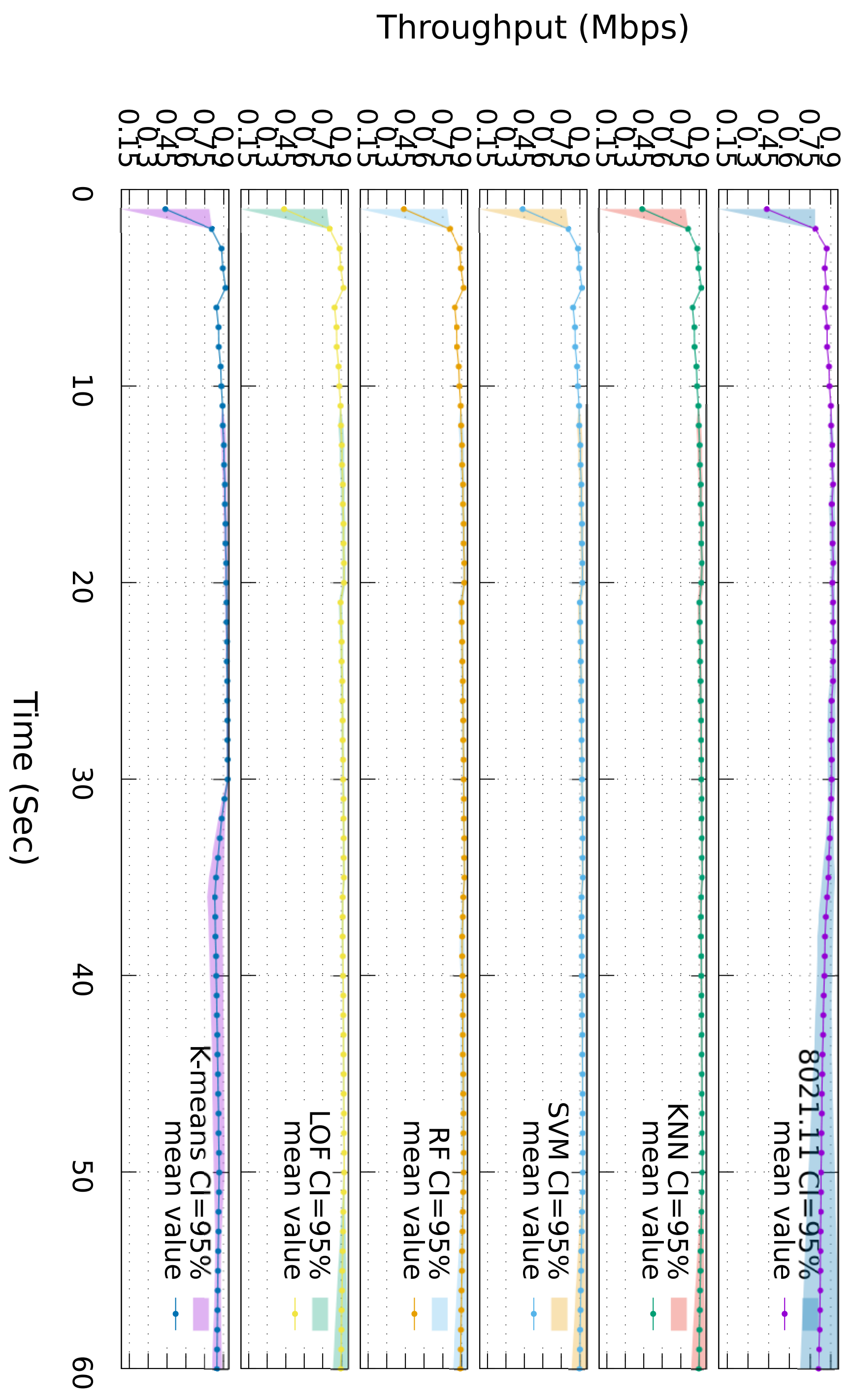


Figure 5.27: Throughput vs. Simulation time when the number AP is changed (Scenario VI)

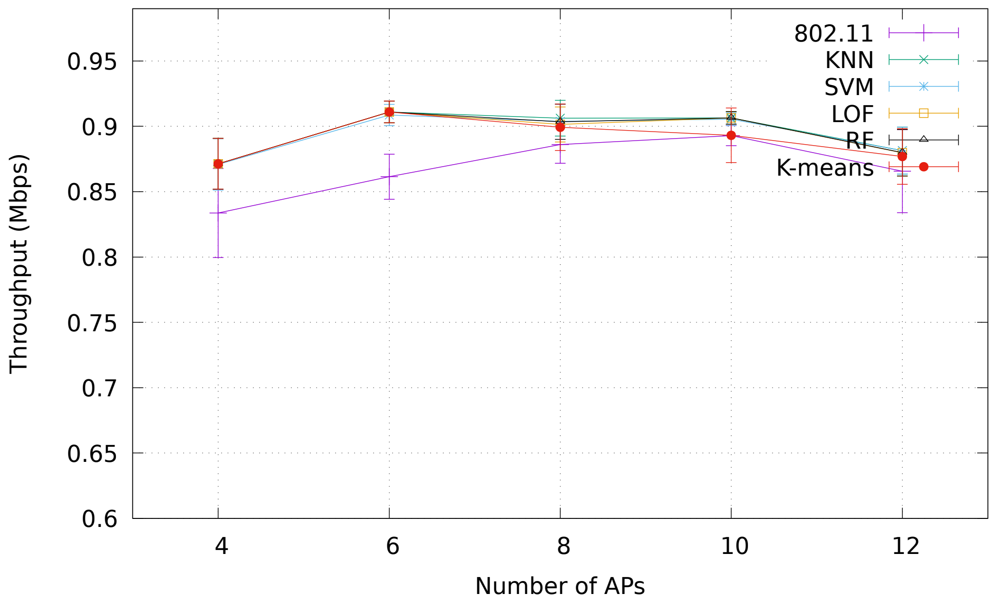


Figure 5.28: Throughput vs. Number of APs (Scenario VI)

ber of APs. Although there is a faulty AP, the high density AP environment means the STA still has more chances to find new APs that can provide better throughput than the current serving AP. The average throughput is not affected too much in high density environment. This is because when the number of AP increases, the distance between APs is reduced. Thus, the STA always moves within an APs' coverage range and can easily handover to neighbor APs. However, the expected communication time between the STA and AP is reduced because of the dense environment. As the proposed self-healing method considers detecting the faulty APs, the throughput is less affected than 802.11.

- **Packet loss rate**

Packet loss happens during handover because of the disruption time. When the AP is faulty, it also causes a high packet loss rate due to the disconnectivity. The packet loss rate also depends strongly on the density of networks. In the results shown in Figure 5.29, the packet loss rate of the machine learning detection methods are compared with the standard 802.11 as the number of APs increase from four to twelve.

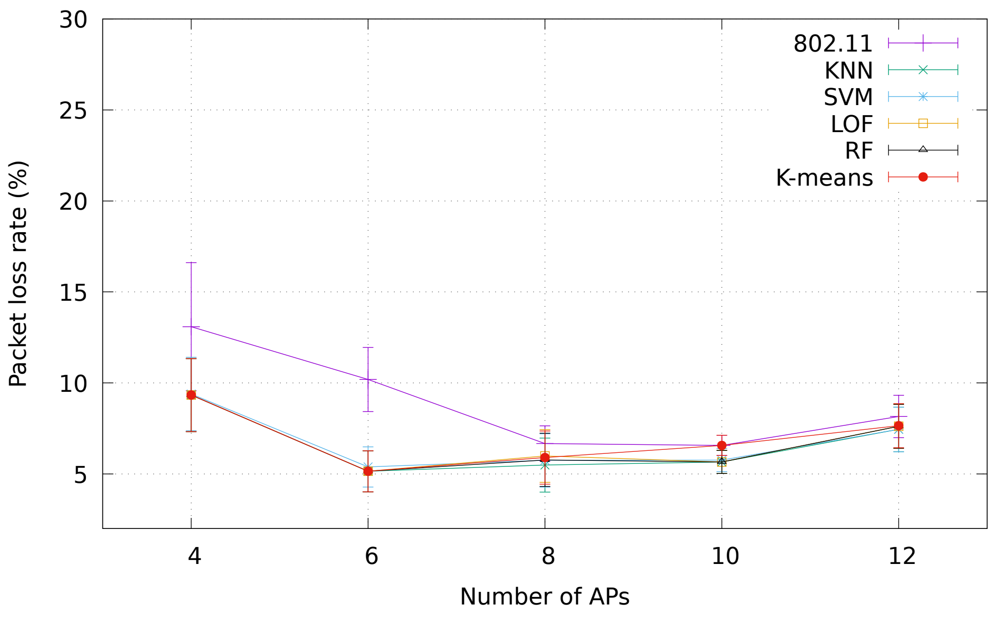


Figure 5.29: Packet loss rate vs. Number of APs (Scenario VI)

When the number of AP increases from four to eight, the packet loss reduces because the STA has more chances to associate with new APs that can provide better service than the current AP. However, when the number of AP continues to increase, the packet loss rate increases a little bit because of the interferences between APs.

With the self-healing for WiFi networks, the average packet loss rate of KNN has been reduced to 6.61%. The packet loss rate of KNN is 25.98% lower than standard 802.11. The packet loss rate of KNN is lower than other algorithms such as SVM and random forest, which means the KNN method is not affected significantly by the AP density changing. When the number of AP is six to ten, the average packet loss is almost the same using KNN, SVM, random forest and LOF. Compared with the standard 802.11, the average packet loss of K-means only reduced by 21.94%. The permanence of K-means becomes worse and worse when the number of AP reaches ten and twelve. In dense AP networks, K-means cannot detect the faulty AP with high accuracy because of its poor scalability. This causes K-means to be unable to detect the faulty AP correctly in a global view. Therefore, the packet loss reduces less than other self-healing algorithms because the low detection accuracy can still cause data disruption.

- **Handover delay**

  Figure 5.30 shows the handover delay vs. the number of APs.

  The average handover delay is lowest for 802.11 when the number of APs is four because the STA loses connection when the outage happens and fails to find better AP to associate with. Thus, there is actually no handover for 802.11, which leads to the lowest total handover delay.

  When the number of AP is six, all six mechanisms achieve almost identical handover delays, which is because the faulty AP3 has not affected the handover performance. The STA associated with AP2 handovers to AP4 directly. When the number of APs is eight, the handover delay is affected by the faulty AP because the STA moves before AP3 and just wants to associate with AP3. As the number of AP increased (10 and 12 APs), KNN and RF achieve better handover performance. This is because KNN detects the faulty AP based on only the K nearest neighbors and increasing the number of APs does not affect the efficiency of the algorithm. RF has no over-fitting problem so it can detect the faulty AP with higher accuracy. The unsupervised learning LOF and K-means cause high handover delay when the number of APs increases to 10 or 12 because the clustering algorithms take longer to detect the AP status when the number of AP increases.

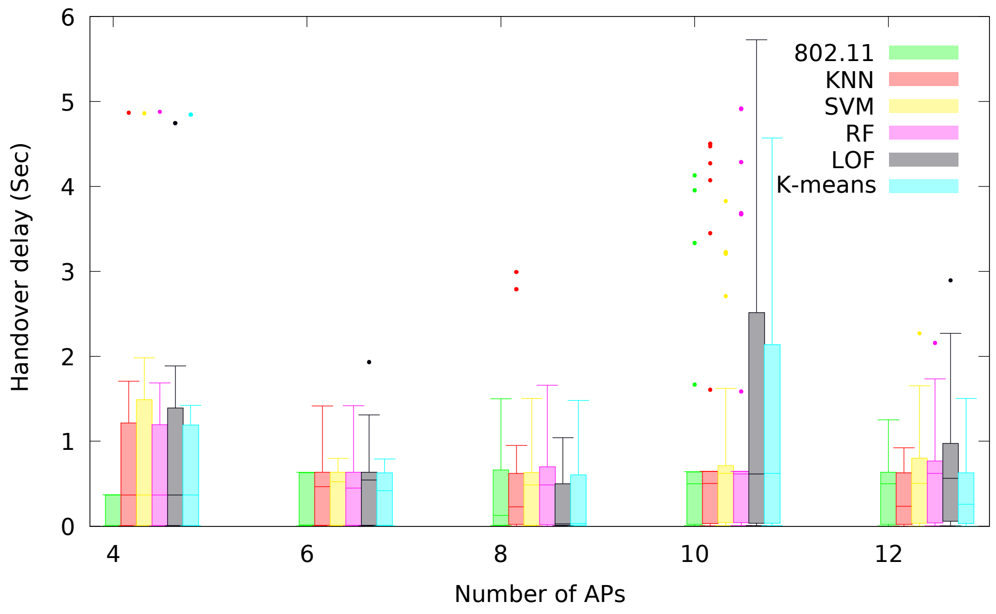


Figure 5.30: Handover delay vs. Number of APs (Scenario VI)

5.4.3.7 Changing the Number of STAs

In Scenario VII, the number of STAs has been increased to see the impact on handover performance. The STAs start to move at t=0s with at a constant speed of 3m/s from AP0 to AP5. The distance between APs is 30m. At t = 25s, AP2 has been set to sleep status and transmit power is set to 0.1dBm to simulate hardware failures for 40 seconds. During the sleep mode, the remaining energy of APs is assumed unchanged. The simulation time is 60s for each run.

- **Throughput**

  Usually, the more STAs utilize the capacity of APs, the higher the network throughput becomes. However, the throughput is not always increased when the number of STAs increases in the WiFi networks exiting faulty AP. This is because the STAs lose connectivity when they want to handover to the faulty APs. The scanning time becomes larger for STAs to find a better AP to associate with, which may cause handover failure.

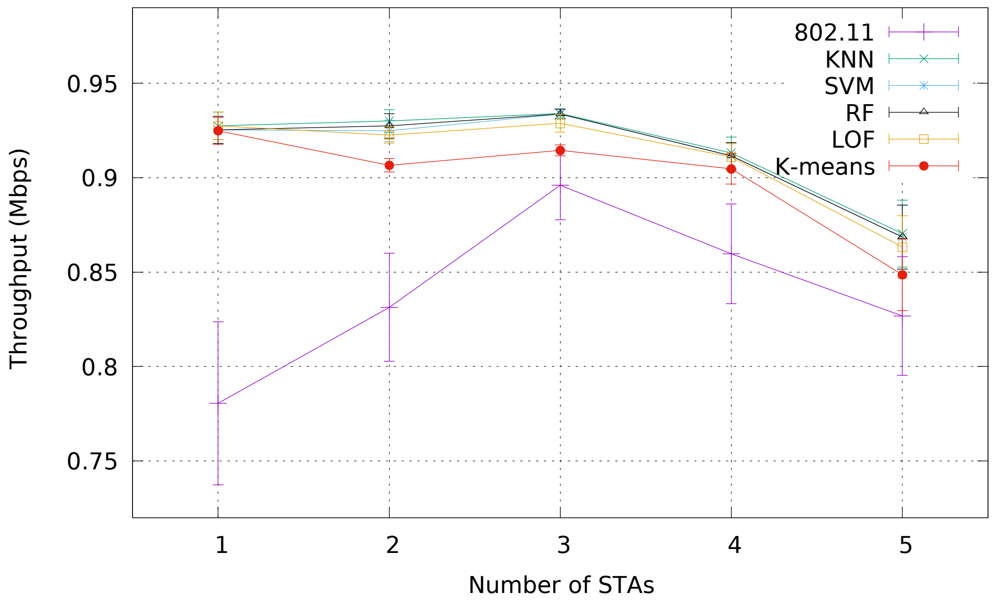


Figure 5.31: Throughput vs. Number of STAs (Scenario VII)

In Figure 5.31, the throughput of five machine learning algorithms including KNN, SVM, RF, LOF, K-means are compared with the 802.11 standard. With the self-healing approach, all the machine learning algorithms show an improvement in throughput. However, the improvement in throughput does not always increase. This is because the larger the number of STAs, the more contention there is among them. Thus, the larger STAs density achieves lower throughput improvement. Compared with the standard 802.11, the average throughput of supervised learning KNN, SVM, and RF have been improved by 9.07%, 8.80% and 8.79% respectively and the average throughput of unsupervised learning LOF and K-means has been improved by 8.67% and 7.26% respectively. The improvement of unsupervised learning algorithms is worse than supervised learning algorithms, especially K-means shows the worst performance in this scenario. This is because K-means needs to run many iterations to find the faulty AP with a high cost of computation time [159]. Although K-means is a simple algorithm, it cannot meet the dynamic network requirements. When the dense of STA increases, the slower detection of faulty AP by K-means cannot ensure the STAs can associate with a new target AP before the fault is detected. Thus, the average throughput using K-means shows the lowest improvement.

- **Packet loss rate**

  When the number of STAs increases, it usually causes a high packet loss rate due to disconnectivity. In the results shown in Figure 5.32, the packet loss rate of the proposed self-healing methods is compared with the standard 802.11 as the number of STAs increases from 1 to 5.

  In the traditional 802.11 networks, the packet loss rate is reduced when the number of STAs increases from one to three, while the packet loss rate is increased when the number of AP is increased to four and five. This is because the STAs cannot find a better AP to associate with when there is a faulty AP. All the STAs try to associate with the same AP when the outage happens. When the AP load is high, the imbalanced load leads to considerable packet loss.

  In Figure 5.32, the packet loss rate of five machine learning algorithms including KNN, SVM, RF, LOF, K-means has been compared with standard 802.11. With the self-healing approach, the packet loss rate is reduced by all the machine learning algorithms. Compared with standard 802.11 , the average packet loss rate of the supervised learning algorithms KNN, SVM, RF, LOF and K-means has been re-

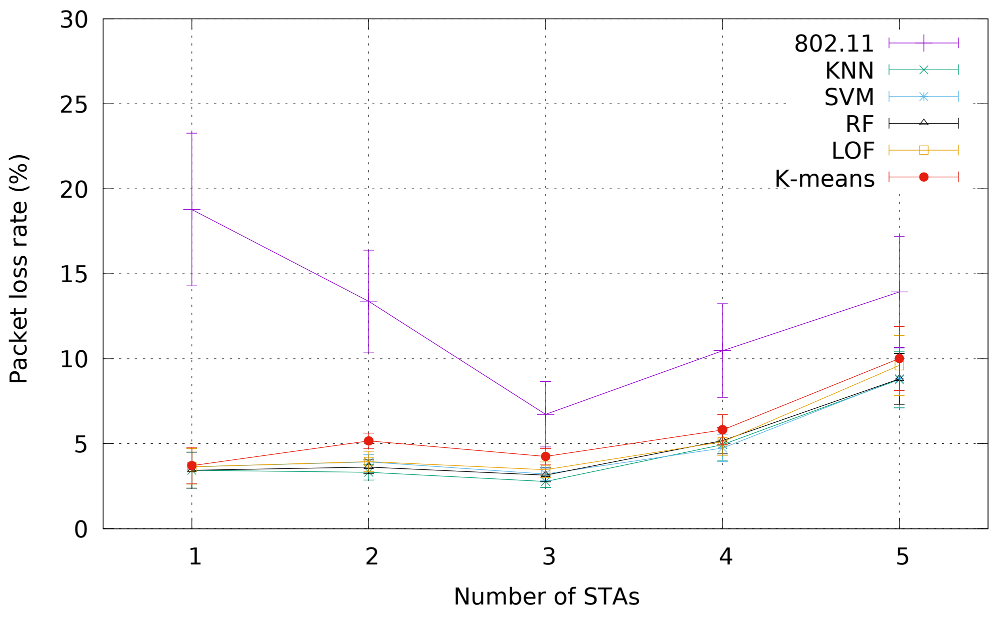


Figure 5.32: Packet loss rate vs. Number of STAs (Scenario VII)

duced by 63.28%, 61.49%, 61.78%, 59.28% and 54.27% respectively. The supervised algorithms show better improvement than unsupervised algorithms. This is because supervised algorithms have lower detection error than unsupervised algorithms. When the STAs obtain the faulty AP's information, they re-associate with a new AP on the neighbor list. The successful handover not only reduces the collision probability among the STAs but also prevent STAs from associating with a faulty AP. Thus, the packet loss rate of supervised algorithms is reduced higher than unsupervised algorithms.

- **Handover delay**

Figure 5.33 shows the average handover delay vs. the number of STAs. As the number of STAs increases, the handover delay is reduced by all self-healing algorithms. The average handover delay is reduced by 39.09% with KNN, 13.63% with SVM, 21.61% with RF and 12.03% with LOF and 13.45% with K-means. KNN achieves the lowest handover delay among the five machine learning algorithms. When the number of STAs increases, the dataset becomes larger. SVM and RF have to spend more time training to update the training model based on the incoming measurement data. LOF and K-means are not sensitive with the less dense AP environment so the faulty AP cannot be detected with higher accuracy. As KNN is only based on the data domain that the incoming measurement data compares with the AP status of K nearest APs without updating the training model, it has less detection time than other algorithms. This efficient learning method reduces the unwanted scanning time of faulty APs and prevented the STAs from associating with a faulty AP. Therefore, the handover delay is reduced.

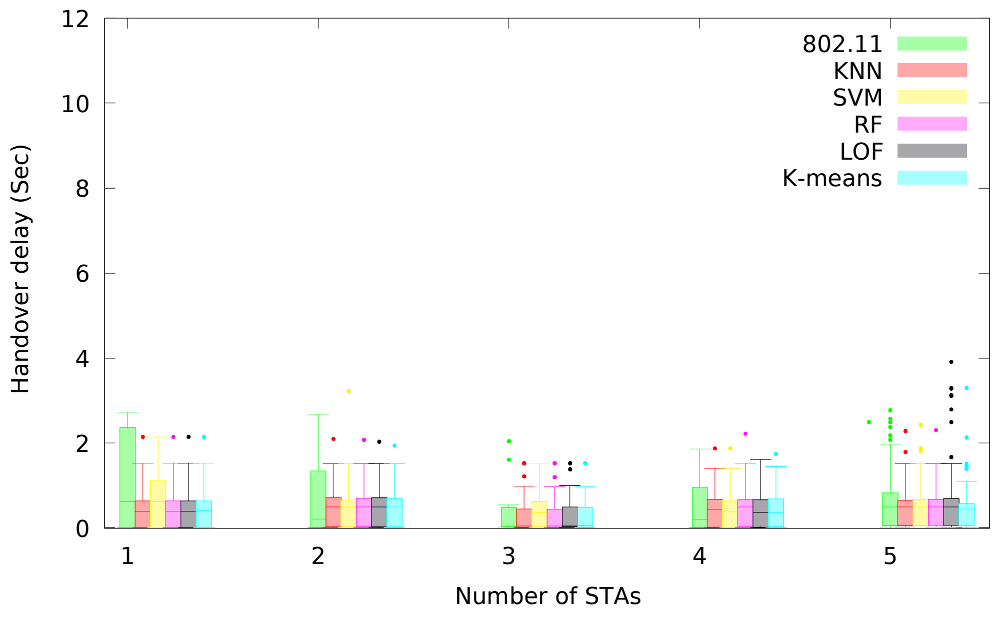


Figure 5.33: Handover delay vs. Number of STAs (Scenario VII)

5.4.3.8 Changing the Network Topology

In Scenario IV, three different topologies of AP placement are investigated. First, the APs are placed in one line as shown in Figure 5.34. The distance between two APs is 30m. One STA moves from AP0 to AP5 begins at t=0s with a constant speed of 3m/s from left to right, then moves backward and forwards between AP5 and AP0. At t = 20s, AP3 and AP4 have been set to sleep status and transmit power is set to 0.1dBm to simulate hardware failures for 80 seconds. During the sleep mode, there is no energy consumed by AP3 and AP4. The simulation time is 100s for each run.

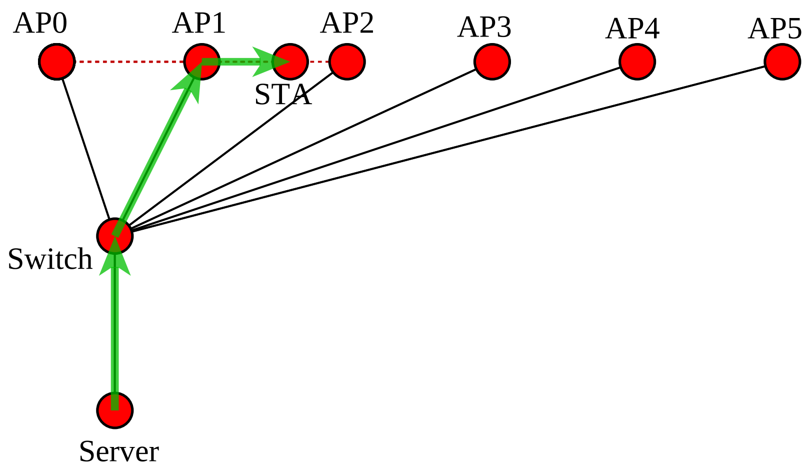


Figure 5.34: AP placement of 1D line topology for self-healing approach

In the second topology, the APs are placed in a grid as shown in Figure 5.35. The distance between the rows is 16m and 12 APs placed on each line. At t=0s, one STA moves in the middle among lines back and forth with a constant speed of 3m/s. The distance between APs is 30m. At t = 20s, AP5 and AP9 have been set to sleep status and transmit power is set to 0.1dBm to simulate hardware failures for 80 seconds. During the sleep mode, there is no energy consumed by AP3 and AP5.

The third topology showed in Figure 5.36 consists of a random AP placement in a 90×80m area. One STA moves with a constant speed of 3m/s from left to right backward and forwards horizontally. At t = 20s, AP3 and AP5 have been set to sleep status and transmit power is set to 0.1dBm to simulate hardware failures for 80 seconds. During the sleep

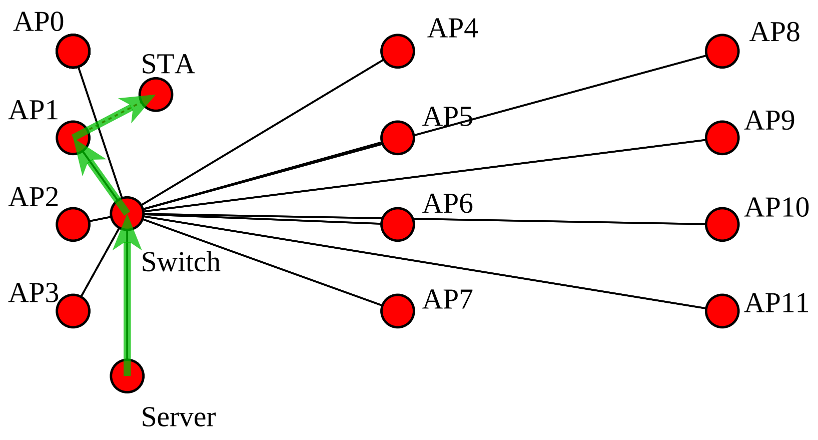


Figure 5.35: AP placement of 2D grid topology for self-healing approach

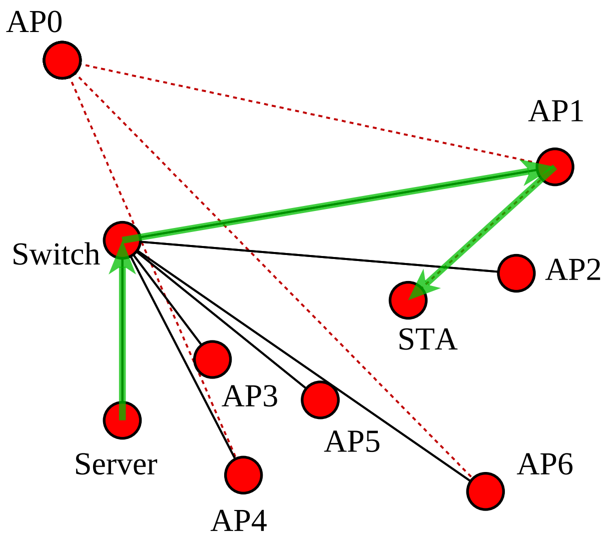


Figure 5.36: AP placement of random topology for self-healing approach

mode, there is no energy consumed by AP3 and AP5.

- **Throughput**

  In Figures 5.37 5.38 and 5.39, the throughput of self-healing algorithms are compared with the 802.11 standard in three different network topologies.

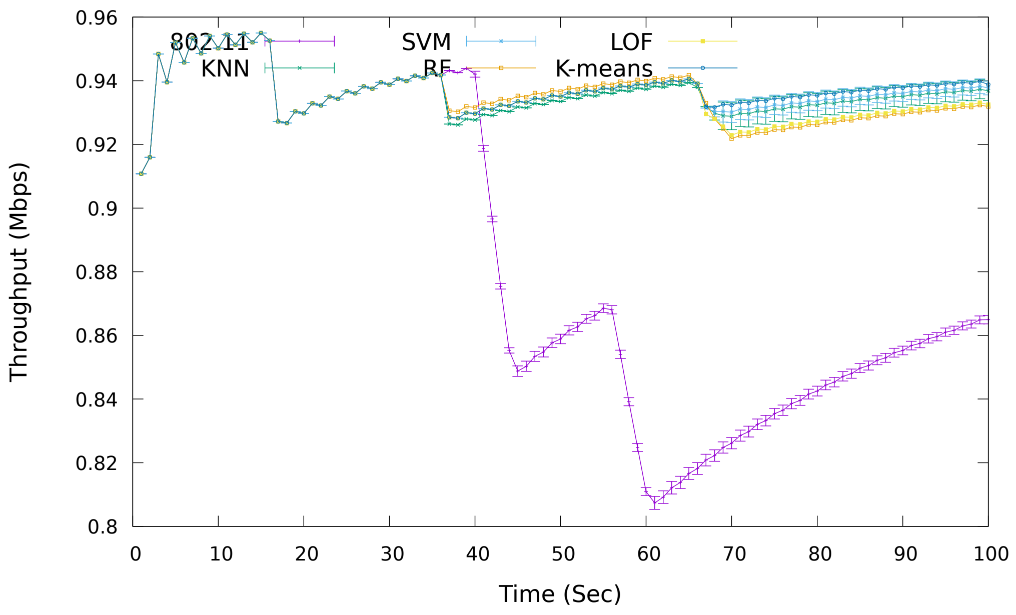


Figure 5.37: Throughput vs. Simulation time - 1D topology (Scenario VIII)

Figure 5.37 shows the average throughput when STA moves in a line network topology. In traditional 802.11 networks, the average throughput drops twice at t = 43s and t =61s because of the faulty AP3 and AP5. At t =43s, the STA associated with AP4 tries to handover to AP5. However, the STA takes a long time to scan for the best candidate AP because both the closest APs to AP4 are faulty APs (AP3 and AP5). At t = 50s, when the STA moves back from AP5, the throughput continues to degrade because the STA scans a long searching for the best candidate AP.

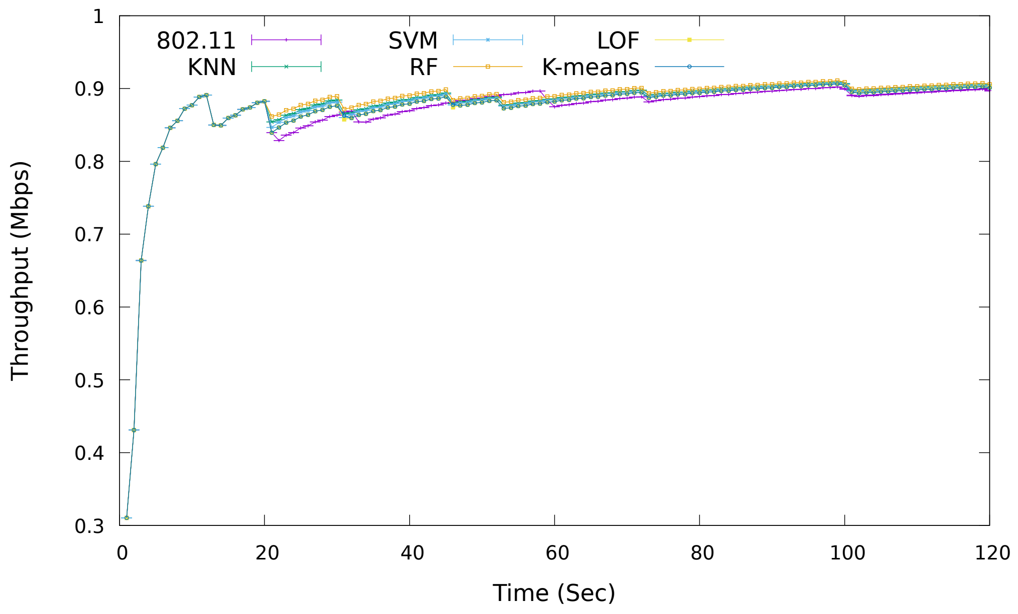


Figure 5.38: Throughput vs. Simulation time - 2D topology (Scenario VIII)

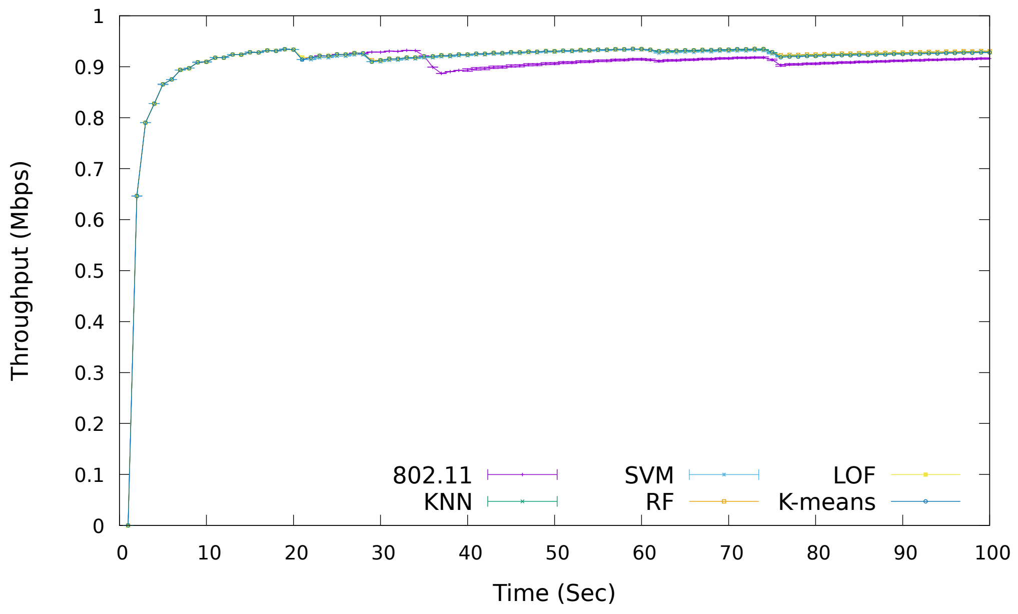


Figure 5.39: Throughput vs. Simulation time - Random topology (Scenario VIII)

Compared with 802.11, the average throughput of self-healing algorithms has been improved by 5.81% with KNN, 5.91% with SVM, 5.71% with RF and 5.68% with LOF and 5.97% with K-means. The performance of K-means shows the best improvement in the line network topology. This is because there are four normal APs and two faulty APs in this line topology networks and it is easier for K-means algorithm to cluster the normal APs and faulty APs in this simple line topology.

Figure 5.38 shows the average throughput when the STA moves in a grid network topology. Compared with standard 802.11, the average throughput of self-healing is improved by 0.78% with KNN, 0.64% with SVM, 1.12% with RF and 0.38% with LOF and 0.37% with K-means.

Compared with 1D line topology, the self-healing algorithms show less improvement to the average throughput in the 2D grid topology. This is because even though there are two faulty APs in this topology, the dense network can keep the data connection the STA can find the best candidate easily because of multiple neighbor APs. However, the self-healing approaches still improved throughput performance. This is because the scanning delay is reduced by informing the STA of the faulty AP in advance. Thus, the STA is prevented from scanning and associating with the faulty APs. The throughput is improved because of the fast handover performance.

Figure 5.39 shows the average throughput when the STA moves in a random network topology. Compared with 802.11, the average throughput of self-healing is improved by 0.82% with KNN, 0.74% with SVM, 0.96% with RF and 0.89% with LOF and 0.87% with K-means.

Compared with the 2D grid topology, the unsupervised algorithms LOF and K-means still show better improvement in this less dense random topology. Thus, the AP density is a key factor to choose self-healing algorithms. If the AP density is low, the unsupervised learning can detect the AP very fast without longer training time. However, supervised learning is more suitable to the dense network environment because the unsupervised clustering algorithms cannot guarantee accuracy when the number of APs is large. If the faulty AP cannot be detected in a real-time network environment, the throughput is unable to be improved because of handover failure.

- **Packet loss rate**

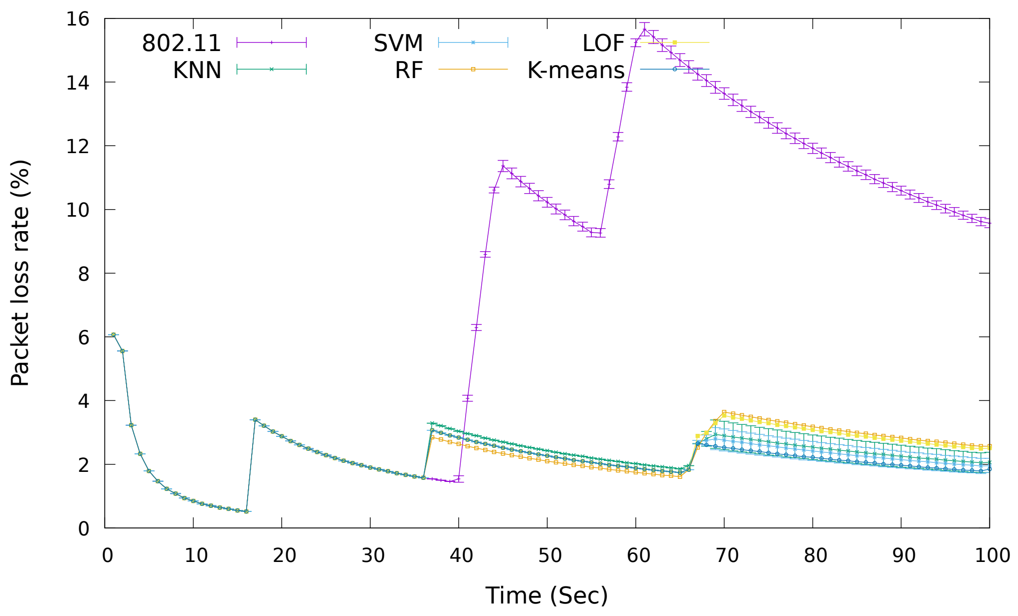


Figure 5.40: Packet loss rate vs. Simulation time - 1D topology (Scenario VIII)

When there are two faulty APs, it usually causes a higher packet loss rate than one faulty AP due to the loss of data connectivity. In the results shown in Figure 5.40 5.41 5.42, the packet loss rate of self-healing methods are compared with 802.11 standard in three network topologies.

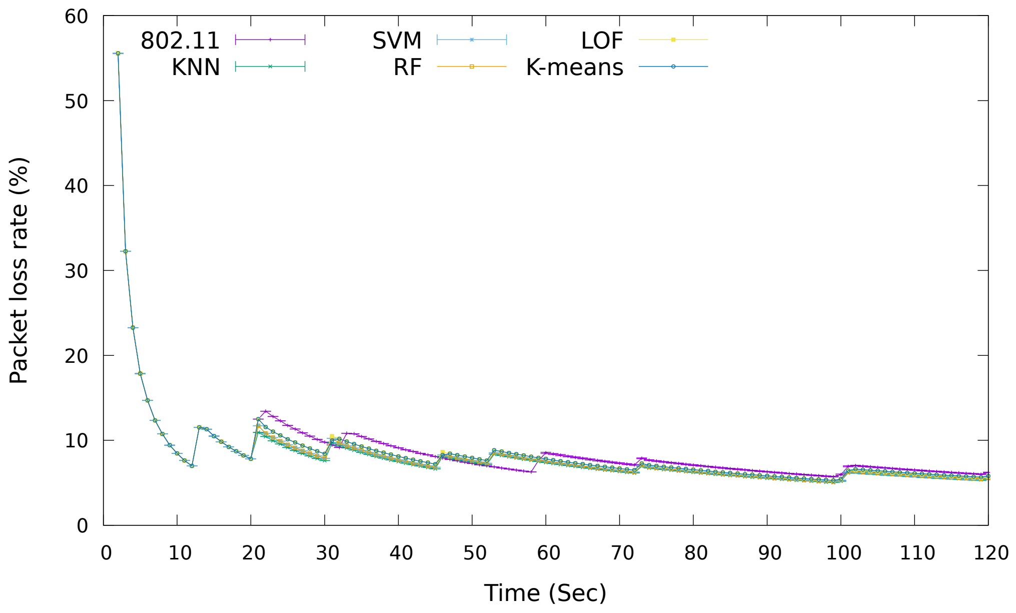


Figure 5.41: Packet loss rate vs. Simulation time - 2D topology (Scenario VIII)

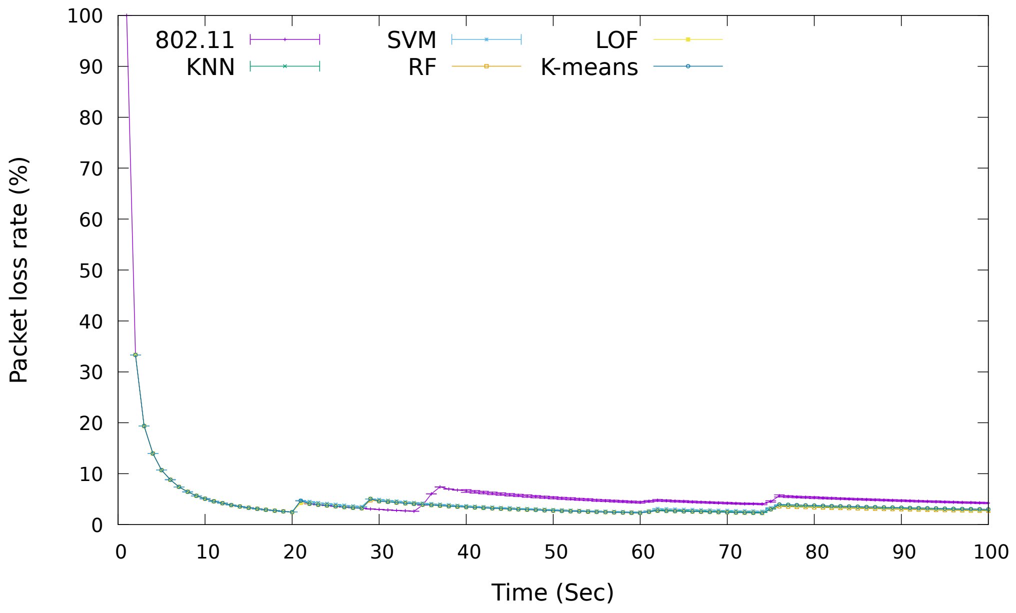


Figure 5.42: Packet loss rate vs. Simulation time - Random topology (Scenario VIII)

The traditional 802.11 standard shows very high average packet loss rate of 15.16%, 10.50% and 7.41% respectively in the 1D, 2D and random topologies when the outage happens. In the 1D line topology, there is no hop between the two faulty APs and the faulty time occurs just before the STA moves to the faulty AP. The distance between the normal AP2 and AP5 is long and the STA scans a long time to search for a new AP to associate with. Thus, it generates a higher packet loss rate compared with the grid topology and the random topology. In this grid topology, the AP placement is dense. Although the two faulty APs are next to each other, the STA can still find a better AP to re-associate with more easily than in the 1D topology. However, the dense network of a grid topology involves more interference between APs. The scanning delay also increases because the number of AP is larger. Thus, the packet loss rate is higher than with the random topology. In the random topology, the APs are placed not too dense. Therefore, the STA can find the best candidate AP more easily than in the 1D line topology because the distance between normal APs is not very far.

With the self-healing approaches, the average packet loss rate has been reduced by 6.88% in the line topology, 6.64% in the grid topology and 2.07% in random. In the 1D topology, the average packet loss rate improved most. This is because two faulty APs cause more data disruption due to the long scanning in traditional 802.11 networks. The self-healing approach detects the faulty AP before the STA moves to the faulty APs so the data disruption time is reduced. The STA performs handover to neighbor APs once they obtain the faulty APs information. Thus, the average packet loss is reduced a lot in the line topology with the use of self-healing algorithms. As the STA does not need to scan the faulty APs in the dense 2D grid topology, the fast handover reduces the packet loss by self-healing algorithms. Thus, the packet loss rate is still reduced a lot.

Compared with 1D and 2D topology, the faulty APs has less effect on packet loss in the random topology. This is because the random topology is a lower AP density so it has fewer interference issues and shorter scanning time than in the 2D topology. Even though two APs are faulty, the STA still can find the best candidate AP when the outage happens. Thus, the packet loss rate is reduced less than 1D line topology. With the self-healing algorithms, the STA reduces the scanning time and the faulty AP is prevented from associating with a faulty AP. Therefore, the packet loss rate is reduced in the random topology networks.

- **Handover delay**

  Figure 5.43 shows the average handover delay vs. different topologies. The handover delay of the 1D topology is higher than the handover delay of the 2D and random topologies. This is because of the longer scanning time caused by two faulty APs in the less dense 1D line topology. The STA is hard to find the best candidate AP because of the long distance between normal APs.

  In the 2D topology, the handover delay is lower than 1D because the number of AP is increased. The STA has more chances to find a new AP to associate with than 1D topology. However, the density of AP causes the STA to scan more APs than the random topology. Thus, the average handover delay is higher than random topology. In the random topology, the AP placement is less dense than 2D topology and the APs are randomly placed not like the line topology. The STA spends less scanning time than 1D and random topology. Thus, the handover delay is the smallest among the three topologies.

  The self-healing algorithms reduce handover delay when the outage happens in the three network topologies. This is because self-healing can dynamically detect the faulty APs based on the real-time network environment. The average handover delay has been reduced by 54.47%, 26.19%, 6.50% in 1D, 2D and random topologies, respectively. In the 1D line topology, the average handover delay is reduced most because the handover delay of 802.11 is the largest in the three topologies. When the faulty AP is detected, the STA can associate with a normal AP on the neighbor list. In the 2D topology, the handover delay has also been reduced in the dense network topology because the STA can find the best candidate easier than 1D topology. The average handover delay of random topology has less affected by the faulty APs because the positions of faulty AP and the faulty time has fewer effects on scanning for the best candidate AP. Thus, the handover delay is reduced less than in the 1D and 2D topologies.

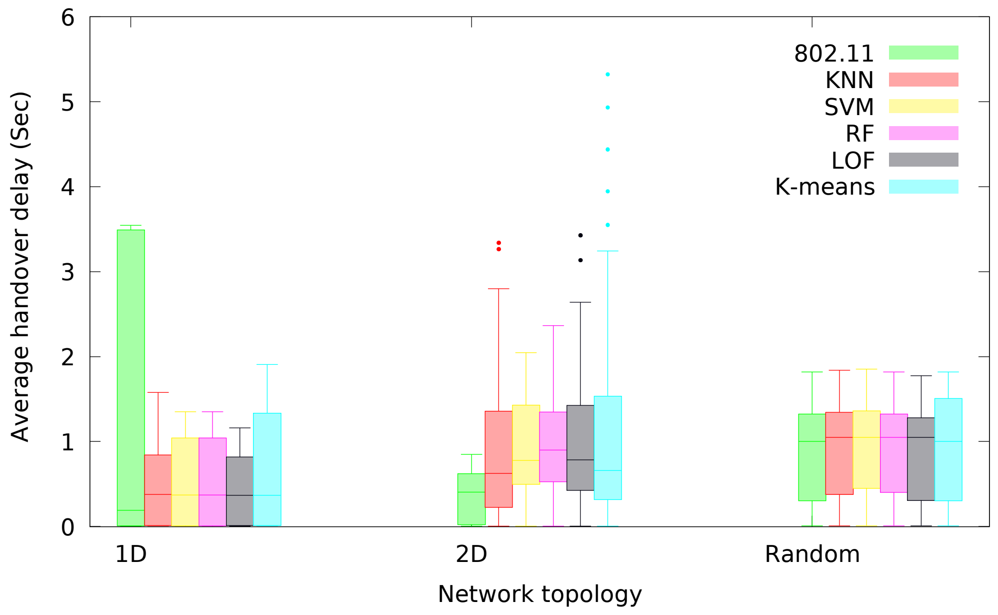


Figure 5.43: Handover delay vs. Handover number in different network topologies (Scenario VIII)

5.5 Summary

This proposed self-healing method for WiFi networks automatically detects the faulty APs and compensates for the network degradation. On the basis of network measurements from APs, the machine learning algorithms KNN, SVM, RF, LOF and K-means have been implemented successfully in the simulation environment. The simulation results demonstrate that the network performance is significantly better than the traditional 802.11 standard.

In the future, how to optimize the transmit power considering AP placement and faulty time to avoid the faulty APs effects on handover performance will be further investigated to provide more practical solutions.

Chapter 6

Conclusions

Self-Organizing Network (SON) is a promising concept for improving network service automatically by reducing operational and maintenance costs. The goal of this thesis is to use SON functionalities to provide seamless handover for WiFi networks. The three SON functionalities used in this thesis are self-configuring neighbor lists, self-optimizing scanning parameters and a self-healing system working together to significantly enhance the handover performance in WiFi networks. Therefore, this thesis contributes to the SON solutions that provide mobility management for WiFi networks taking into consideration the major challenges discussed in Chapter 2.

In Chapter 3, a self-configuring Neighbor List Mechanism (NLM) is proposed. A centralized AP controller (APC) is used to store the up-to-date neighbor lists and manage the AP cooperation. The neighbor APs' information is obtained based on the neighbor reports and the radio parameters retrieved from radio measurement reports provided in standard 802.11k. Compared with the 802.11 standard, the average handover delay using NLM is reduced by approximately 41.46%, the average throughput using NLM is improved by approximately 5.10%, and the average packet loss rate using NLM is reduced by approximately 36.14%.

However, NLM uses the fixed scanning parameters which also affect the scanning delay. In response to this, the scanning parameters were optimized in Chapter 4 to further reduce the unnecessary waiting time (around 24% compared to NLM) of probe responses from non-adjacent APs or APs with poor QoS. Thus, compared with NLM, the average handover delay is reduced by approximately 10.88%, the average throughput is improved by approximately 2.83% and the average packet loss rate is reduced by approximately 12.44%.

The SON functionalities in Chapters 3 and 4 reduce the scanning delay based on the assumption that the network elements all work normally. However, it is inevitable that an AP may suddenly be faulty during the handover process. This may lead to a handover failure. Faulty APs can also cause network degradation. In Chapter 5, a self-healing system is implemented to automatically detect faulty APs and compensate for the network degradation. Classification and clustering algorithms are used for the self-detection phase to find the faulty APs. In self-healing, compared with the 802.11 standard, the average handover delay is reduced by approximately 71.86%, the average throughput is improved by approximately 30.53%, the average packet loss rate is reduced by approximately 68.27%.

6.1 Contributions

The contributions of this thesis are summarized as follows:

1. An up-to-date NLM is designed and evaluated to establish neighborhood management in WiFi networks. This NLM is the first solution for establishing neighborhood relationships under dynamic network conditions using machine learning algorithms based on the IEEE 802.11k standard. The number of channels and APs is reduced during handover process. Therefore, the total scanning delay is reduced, in turn reducing the handover delay in WiFi networks.

2. In NLM, the handover decision is made by the AP score which is a dynamic handover threshold. The AP score calculated using machine learning algorithms is updated based on radio parameters and network parameters including RSSI, SNR, packet delay, packet loss, data rate, average throughput and AP load in the real-time network. This is the first time that the dynamic threshold using more than one predictor has been used to make a handover decision. Therefore, handover performance is improved because of the dynamic handover decision criterion.

3. A self-optimizing scanning algorithm is used to optimize scanning parameters. This algorithm is the first solution to reduce the scanning delay by optimizing *all* the scanning parameters together using a Genetic Algorithm (GA). *All* the scanning parameters including *MinChannelTime*, *MaxChannelTime*, probe request interval, channel switching time , the number of channels scanned and channel sequence are dynamically adjusted based on real-time network conditions.

4. A new self-healing system is designed for fault detection and compensation for network degradation in WiFi networks. This is the first time SON functionalities are used together to automatically detect and compensate for network failures to improve handover performance in WiFi networks. Self-healing includes self-detection of faulty APs and self-compensating for network degradation. When a faulty AP is detected in the self-detection phase, the neighbor list will be updated with the removal of faulty APs. The STAs search for the best candidate AP to perform handover using the optimized scanning parameters. Thus, self-configuring, self-optimizing and self-healing functionalities work together to reduce handover delay and handover failure rate in WiFi networks.

6.2 Practical Implications

The SON functionalities continuously improve handover performance and the seamless handover increases customer satisfaction. As the SON approach does not need extra hardware and human intervention, the cost and time of operation are reduced by this automation technique.

The practical implications of this SON approach are discussed as follows:

1. This SON approach is implemented based on machine learning techniques. Machine learning is a data analytics technique. The output is predicted by the algorithm learning from the data. The data collected in this study is through simulation by NS-3. The results by using different simulation tools or real measurement need to be tested and validated using this SON approach.

2. The data collection of this SON approach is based on multiple scenarios including dense network environment and different network topologies. Moreover, this SON approach aims to meet different application requirements. Therefore, the traffic is generated randomly to simulate different applications including voice call, video calls, video streaming, online gaming. In the system design and planning, large network topologies and dense network environments need to be supported. Furthermore, the data should be collected by running multiple applications in different network environments.

3. This SON approach uses a centralized architecture to develop novel solutions for handover in WiFi networks. A centralized controller is used to manage the network elements, establish and update neighborhood information, optimize scanning parameters and monitor AP status for the fault management. Therefore, this SON approach is only suitable for centralized networks.

6.3 Future Work

There are still some limitations in terms of the scope of this research. This section gives some directions for future study of each SON functionality in WiFi networks. In the future work, the real measurement can be used to validate the accuracy of mathematical analysis and simulation results.

6.3.1 Self-configuration

1. **The neighbor list storage**: The current neighbor lists are stored in the APC under the assumption that there is no link delay between the APC, APs and STAs. In the real-world applications, the link delay which may increase the handover delay should be considered. One suggestion to reduce the link delay is to store the neighbor list on the STAs. As STAs adapt to local changes rapidly, the neighbor list can be obtained and updated immediately when scanning is initiated by a STA.

   However, AP is vendor-specific and does not allow for easy coordination among equipment from different infrastructure vendors. Therefore, it is still a challenge to establish a neighborhood without a centralized controller. In future, the neighbor list stored in STA needs to be compared with the neighbor list stored in APC.

2. **Initial configuration for newly deployment**: The proposed NLM is only for enhancing the handover performance. However, self-configuration can also be used for the initial configuration of newly deployed APs. Currently, the initial radio and network parameters of APs are defined by different operator and vendors. These initial parameters would be dynamically configured by remote and intelligent firmware management to reduce human intervention using self-configuration in WiFi networks.

6.3.2 Self-optimization

1. **Fitness function in GA**: The current fitness function of GA only considers the candidate AP priority based on SNR, which means the optimal scanning parameters are selected mainly depending on SNR. As the fitness function is the key point to find the optimal scanning parameters, the network parameters used in the fitness function may affect the value of the optimal scanning parameters. In the future work, other radio and network parameters including RSSI, packet delay, packet loss, data rate, throughput and AP load should be considered to meet the requirements of different applications.

2. **Crossover rate and mutation rate of GA**: The crossover rate and mutation rate values are fixed in this approach. The fixed crossover and mutation rate cannot fit the dynamic network environment. As the crossover and mutation affect the algorithm computation time, if the chromosomes do not need to be changed, it is a waste of time to do these operations. Therefore, it is necessary to find a solution to dynamically adjust crossover and mutation rates based on real-time network conditions.

6.3.3 Self-healing

1. **Faulty AP detecting**: In self-healing, the remaining energy is used to detect the faulty AP. Sometimes, an AP without energy consumption might not be a faulty AP such as a sleep AP. Therefore, other effects including the signal strength, traffic and load balance need to be considered to monitor and detect the faulty AP using different network parameters. These network parameters can be RSSI, SNR, packet delay, packet loss, data rate, throughput and AP load. In NS3, low transmit power and sleep AP are used to simulate hardware failures. In future, the real measurement needs to be used for detecting the real AP status.

2. **Dynamically adjusting transmit power**:

   In the current self-healing, network degradation cannot be recovered if the STA is out of the normal AP's coverage range when the large number of faulty AP happens. The neighbor information may be missing because of the limitation of the coverage range. In this situation, the transmit power of normal APs are manually increased to a higher value. The STA can find the best AP to reassociate with when the neighbor information is obtained because of the increased transmit power. However, this method cannot satisfy the dynamic network requirements. Therefore, the transmit power should be dynamically adjusted to meet the network requirements when the outage happens.

Bibliography

[1] M. S. Bargh, R. Hulsebosch, E. Eertink, A. Prasad, H. Wang, and P. Schoo, “Fast authentication methods for handovers between ieee 802.11 wireless lans,” in *Proceedings of the 2nd ACM international workshop on Wireless mobile applications and services on WLAN hotspots*, pp. 51–60, ACM, 2004.

[2] L. Hao, B. Ng, and Y. Qu, “Dynamic optimization of neighbor list to reduce changeover latency for wi-fi networks,” in *Proceedings of the 2017 International Conference on Telecommunications and Communication Engineering*, ICTCE ’17, (New York, NY, USA), pp. 20–24, ACM, 2017.

[3] M. Shin, A. Mishra, and W. A. Arbaugh, “Improving the latency of 802.11 hand-offs using neighbor graphs,” in *Proceedings of the 2Nd International Conference on Mobile Systems, Applications, and Services*, MobiSys ’04, (New York, NY, USA), pp. 70–83, ACM, 2004.

[4] S. Shin, A. G. Forte, A. S. Rawat, and H. Schulzrinne, “Reducing mac layer handoff latency in ieee 802.11 wireless lans,” in *Proceedings of the Second International Workshop on Mobility Management &Amp; Wireless Access Protocols*, MobiWac ’04, (New York, NY, USA), pp. 19–26, ACM, 2004.

[5] C.-C. Tseng, K.-H. Chi, M.-D. Hsieh, and H.-H. Chang, "Location-based fast handoff for 802.11 networks," *IEEE Communications Letters*, vol. 9, pp. 304–306, April 2005.

[6] G. Castignani, A. Arcia, and N. Montavont, "A study of the discovery process in 802.11 networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 15, no. 1, pp. 25–36, 2011.

[7] R. W. Pazzi, Z. Zhang, and A. Boukerche, "Performance evaluation of a fast mac handoff scheme using dynamic adjustment of scanning parameters," in *Proceedings of the 12th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, pp. 346–352, ACM, 2009.

[8] A. Boukerche, Z. Zhang, and R. W. Pazzi, "A self-configured handoff scheme for ieee 802.11-based wireless networks," in *Local Computer Networks, 2009. LCN 2009. IEEE 34th Conference on*, pp. 124–129, IEEE, 2009.

[9] A. A. Tabassam, H. Trsek, S. Heiss, and J. Jasperneite, "Fast and seamless handover for secure mobile industrial applications with 802.11 r," in *2009 IEEE 34th Conference on Local Computer Networks*, pp. 750–757, IEEE, 2009.

[10] P. Muñoz, R. Barco, and I. de la Bandera, "On the potential of handover parameter optimization for self-organizing networks," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 5, pp. 1895–1905, 2013.

[11] Y. Lee, B. Shin, J. Lim, and D. Hong, "Effects of time-to-trigger parameter on handover performance in son-based lte systems," in *2010 16th Asia-Pacific Conference on Communications (APCC)*, pp. 492–496, IEEE, 2010.

[12] W. Zheng, H. Zhang, X. Chu, and X. Wen, "Mobility robustness optimization in self-organizing lte femtocell networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 1, p. 27, 2013.

[13] A. Lobinger, S. Stefanski, T. Jansen, and I. Balan, "Coordinating handover parameter optimization and load balancing in lte self-optimizing networks," in *2011 IEEE 73rd vehicular technology conference (VTC Spring)*, pp. 1–5, IEEE, 2011.

[14] I. N. M. Isa, M. D. Baba, A. L. Yusof, and R. A. Rahman, "Handover parameter optimization for self-organizing lte networks," in *Computer Applications Industrial Electronics (ISCAIE), 2015 IEEE Symposium on*, pp. 1–6, April 2015.

[15] "Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pp. 1–2793, March 2012.

[16] "Ieee standard for telecommunications and information exchange between systems - lan/man specific requirements - part 11: Wireless medium access control (mac) and physical layer (phy) specifications: High speed physical layer in the 5 ghz band," *IEEE Std 802.11a-1999*, pp. 1–102, Dec 1999.

[17] "Ieee standard for information technology - telecommunications and information exchange between systems - local and metropolitan networks - specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications: Higher speed physical layer (phy) extension in the 2.4 ghz band - corrigendum 1," *IEEE Std 802.11b-1999/Cor 1-2001*, pp. 1–24, Nov 2001.

[18] "Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications: Further higher data rate extension in the 2.4 ghz band," *IEEE Std 802.11g-2003 (Amendment to IEEE Std 802.11, 1999 Edn. (Reaff 2003) as amended by IEEE Stds 802.11a-1999, 802.11b-1999, 802.11b-1999/Cor 1-2001, and 802.11d-2001)*, pp. 1–104, June 2003.

[19] "Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless lan medium access control (mac)and physical layer (phy) specifications amendment 1: Radio resource measurement of wireless lans," *IEEE Std 802.11k-2008 (Amendment to IEEE Std 802.11-2007)*, pp. 1–244, June 2008.

[20] "Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 2: Fast basic service set (bss) transition," *IEEE Std 802.11r-2008 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008)*, pp. 1–126, July 2008.

[21] I. Prasetyo, M. Anif, A. S. Nugroho, S. Pramono, S. Widodo, and S. S. Hidayat, "Handover analysis of data and voip services in 802.11 b/g/n wireless lan," *TELKOMNIKA Indones. J. Electr. Eng*, vol. 12, no. 11, pp. 7832–7844, 2014.

[22] A. Böhm and M. Jonsson, "Handover in ieee 802.11 p-based delay-sensitive vehicle-to-infrastructure communication," 2009.

[23] Z. Zhang, W. P. Richard, and A. Boukerche, "A fast mac layer hand-off protocol for wifi-based wireless networks," in *Local computer networks (LCN), 2010 IEEE 35th conference on*, pp. 684–690, IEEE, 2010.

[24] A. R. Rebai and S. Hanafi, "An adaptive multimedia-oriented handoff scheme for ieee 802.11 wlans," *arXiv preprint arXiv:1102.5189*, 2011.

[25] A. Mishra, M. Shin, and W. A. Arbaush, "Context caching using neighbor graphs for fast handoffs in a wireless network," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 1, p. 361, March 2004.

[26] A. Mishra, M. Shin, and W. Arbaugh, "An empirical analysis of the ieee 802.11 mac layer handoff process," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, pp. 93–102, 2003.

[27] H. Velayos and G. Karlsson, "Techniques to reduce the ieee 802.11 b handoff time," in *Communications, 2004 IEEE International Conference on*, vol. 7, pp. 3844–3848, IEEE, 2004.

[28] S. Shin, A. G. Forte, A. S. Rawat, and H. Schulzrinne, "Reducing mac layer handoff latency in ieee 802.11 wireless lans," in *Proceedings of the second international workshop on Mobility management & wireless access protocols*, pp. 19–26, ACM, 2004.

[29] A. Arcia-Moret, A. Araujo, J. Aguilarz, L. Molinax, and A. Sathiaseelan, "Intelligent network discovery for next generation community wireless networks," in *Wireless On-demand Network Systems and Services (WONS), 2016 12th Annual Conference on*, pp. 1–7, IEEE, 2016.

[30] I. Ramani and S. Savage, "Syncscan: practical fast handoff for 802.11 infrastructure networks," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 1, pp. 675–684 vol. 1, March 2005.

[31] D. Mills, "Network time protocol (version 3) specification, implementation," 1992.

[32] Y.-S. Chen, C.-K. Chen, and M.-C. Chuang, "Deucescan: Deuce-based fast handoff scheme in ieee 802.11 wireless networks," in *VTC*, 2006.

[33] V. M. Chintala and Q. A. Zeng, "Novel mac layer handoff schemes for ieee 802.11 wireless lans," in *2007 IEEE Wireless Communications and Networking Conference*, pp. 4435–4440, March 2007.

[34] B. Ng, A. Deng, Y. Qu, and W. K. Seah, "Changeover prediction model for improving handover support in campus area wlan," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, pp. 265–272, IEEE, 2016.

[35] V. Mhatre and K. Papagiannaki, "Using smart triggers for improved user performance in 802.11 wireless networks," in *Proceedings of the 4th international conference on Mobile systems, applications and services*, pp. 246–259, ACM, 2006.

[36] A. Bacioccola, C. Cicconetti, and G. Stea, "User-level performance evaluation of voip using ns-2," in *Proceedings of the 2nd international conference on Performance evaluation methodologies and tools*, p. 20, ICST (Institute for Computer Sciences, Social-Informatics and , 2007.

[37] K. Tsukamoto, T. Yamaguchi, S. Kashihara, and Y. Oie, "Experimental evaluation of decision criteria for wlan handover: Signal strength and frame retransmission," *IEICE transactions on communications*, vol. 90, no. 12, pp. 3579–3590, 2007.

[38] G. Athanasiou, T. Korakis, O. Ercetin, and L. Tassiulas, "Dynamic cross-layer association in 802.11-based mesh networks," in *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*, pp. 2090–2098, IEEE, 2007.

[39] S.-J. Yoo, D. Cypher, and N. Golmie, "Timely effective handover mechanism in heterogeneous wireless networks," *Wireless Personal Communications*, vol. 52, no. 3, pp. 449–475, 2010.

[40] A. Bagwari and G. S. Tomar, "Adaptive double-threshold based energy detector for spectrum sensing in cognitive radio networks," *International Journal of Electronics Letters*, vol. 1, no. 1, pp. 24–32, 2013.

[41] H. Gacanin and A. Ligata, "Wi-fi self-organizing networks: Challenges and use cases," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 158–164, 2017.

[42] S. Feng and E. Seidel, "Self-organizing networks (son) in 3gpp long term evolution," *Nomor Research GmbH, Munich, Germany*, vol. 20, 2008.

[43] H. Zhou, "A dynamic neighbor cell list generating algorithm in cellular system," in *2009 5th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1–4, IEEE, 2009.

[44] D. Kim, B. Shin, D. Hong, and J. Lim, "Self-configuration of neighbor cell list utilizing e-utran nodeb scanning in lte systems," in *2010 7th IEEE Consumer Communications and Networking Conference*, pp. 1–5, IEEE, 2010.

[45] 3GPP, "Digital cellular telecommunications system (phase 2+); base station controller - base transceiver station (bsc- bts) interface; interface principles," *3GPP TS 48.052*, vol. V8.0.0, January, 2009.

[46] 3GPP, "3rd generation partnership project; technical specification group radio access network; utran iub interface: General aspects and principles," *3GPP TS 25.430*, vol. V3.8.0, June, 2002.

[47] 3GPP, "Lte; access network (e-utran); x2 application protocol (x2ap)," *3GPP TS 136.423*, vol. V11.2.0, October, 2012.

[48] 3GPP, "Telecommunication management; automatic neighbo bour relation (anr) managemement; concepts and requirements," *3GPP TS 32.500*, vol. v13.0.0, Feb, 2016.

[49] A. Mishra, M. H. Shin, N. L. Petroni, T. C. Clancy, and W. A. Arbaugh, "Proactive key distribution using neighbor graphs," *IEEE Wireless communications*, vol. 11, no. 1, pp. 26–36, 2004.

[50] "Draft ieee recommended practice for multi-vendor access point interoperability via an inter-access point protocol across distribution systems supporting ieee 802.11 operation (superseded by 802.11f-2003)," *IEEE Std P802.11F/D6*, 2003.

[51] F. Parodi, M. Kylvaja, G. Alford, J. Li, and J. Pradas, "An automatic procedure for neighbor cell list definition in cellular networks," in *2007 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pp. 1–6, IEEE, 2007.

[52] Y. Zhang, Y. Liu, Y. Xia, and Q. Huang, "Leapfrog: Fast, timely wifi handoff," in *IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference*, pp. 5170–5174, IEEE, 2007.

[53] H. Zhang, Z. Lu, X. Wen, and Z. Hu, "Qoe-based reduction of handover delay for multimedia application in ieee 802.11 networks," *IEEE Communications Letters*, vol. 19, no. 11, pp. 1873–1876, 2015.

[54] B. Zhang, X. Wen, Z. Lu, T. Lei, and X. Zhao, "A fast handoff scheme for ieee 802.11 networks using software defined networking," in *2016 19th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pp. 476–481, IEEE, 2016.

[55] S. Pack, H. Jung, T. Kwon, and Y. Choi, "A selective neighbor caching scheme for fast handoff in ieee 802.11 wireless networks," in *IEEE International Conference on Communications, 2005. ICC 2005. 2005*, vol. 5, pp. 3599–3603, IEEE, 2005.

[56] T. Lei, X. Wen, Z. Lu, W. Jing, B. Zhang, and G. Cao, "Handoff management scheme based on frame loss rate and rssi prediction for ieee 802.11 networks," in *2016 International Symposium on Wireless Communication Systems (ISWCS)*, pp. 555–559, IEEE, 2016.

[57] D. Aziz, A. Ambrosy, L. T. Ho, L. Ewe, M. Gruber, and H. Bakker, "Autonomous neighbor relation detection and handover optimization in lte," *Bell Labs Technical Journal*, vol. 15, no. 3, pp. 63–83, 2010.

[58] V. Capdevielle, A. Feki, and A. Fakhreddine, "Self-optimization of handover parameters in lte networks," in *2013 11th International Symposium and Workshops on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, pp. 133–139, IEEE, 2013.

[59] P. Bhattacharya, A. Sarkar, S. Chatterjee, *et al.*, "An ann based call handoff management scheme for mobile cellular network," *arXiv preprint arXiv:1401.2230*, 2014.

[60] M. Ekpenyong, J. Isabona, and E. Isong, "Handoffs decision optimization of mobile celular networks," in *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 697–702, IEEE, 2015.

[61] M. S. Dang, A. Prakash, D. K. Anvekar, D. Kapoor, and R. Shorey, "Fuzzy logic based handoff in wireless networks," in *VTC2000-Spring. 2000 IEEE 51st Vehicular Technology Conference Proceedings (Cat. No. 00CH37026)*, vol. 3, pp. 2375–2379, IEEE, 2000.

[62] S. S. Mwanje and A. Mitschele-Thiel, "Distributed cooperative q-learning for mobility-sensitive handover optimization in lte son," in *2014 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, IEEE, 2014.

[63] I. Balan, T. Jansen, B. Sas, I. Moerman, and T. Kürner, "Enhanced weighted performance based handover optimization in lte," in *2011 Future Network & Mobile Summit*, pp. 1–8, IEEE, 2011.

[64] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations of genetic algorithms*, vol. 1, pp. 69–93, Elsevier, 1991.

[65] J. H. Holland *et al.*, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[66] B. A. Julstrom, "It's all the same to me: Revisiting rank-based probabilities and tournaments," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 2, pp. 1501–1505, IEEE, 1999.

[67] R. Oladele and J. Sadiku, "Genetic algorithm performance with different selection methods in solving multi-objective network design problem," *International Journal of Computer Applications*, vol. 70, no. 12, 2013.

[68] J. Zhong, X. Hu, J. Zhang, and M. Gu, "Comparison of performance between different selection strategies on simple genetic algorithms," in *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, vol. 2, pp. 1115–1121, IEEE, 2005.

[69] H. M. Pandey, "Performance evaluation of selection methods of genetic algorithm and network security concerns," *Procedia Computer Science*, vol. 78, pp. 13–18, 2016.

[70] E. D. Goodman, "Introduction to genetic algorithms," in *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pp. 205–226, ACM, 2014.

[71] A. D. Irfianti, R. Wardoyo, S. Hartati, and E. Sulistyoningsih, "Determination of selection method in genetic algorithm for land suitability," in *MATEC Web of Conferences*, vol. 58, p. 03002, EDP Sciences, 2016.

[72] P. Sharma and A. Wadhwa, "Analysis of selection schemes for solving an optimization problem in genetic algorithm," *International Journal of Computer Applications*, vol. 93, no. 11, 2014.

[73] O. Al Jadaan, L. Rajamani, and C. Rao, "Improved selection operator for ga.," *Journal of Theoretical & Applied Information Technology*, vol. 4, no. 4, 2008.

[74] J. E. Baker, "Adaptive selection methods for genetic algorithms," in *Proceedings of an International Conference on Genetic Algorithms and their applications*, pp. 101–111, Hillsdale, New Jersey.

[75] G. Pavai and T. Geetha, "A survey on crossover operators," *ACM Computing Surveys (CSUR)*, vol. 49, no. 4, p. 72, 2017.

[76] T. Weise, "Global optimization algorithms-theory and application," *Self-Published Thomas Weise*, 2009.

[77] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Evolutionary computation 1: Basic algorithms and operators*, vol. 1. CRC press, 2000.

[78] K. Deep and M. Thakur, "A new mutation operator for real coded genetic algorithms," *Applied mathematics and Computation*, vol. 193, no. 1, pp. 211–230, 2007.

[79] Z. Michalewicz and S. J. Hartley, "Genetic algorithms+ data structures= evolution programs," *Mathematical Intelligencer*, vol. 18, no. 3, p. 71, 1996.

[80] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine learning*, vol. 3, no. 2, pp. 95–99, 1988.

[81] Z. Michalewicz, "Evolution strategies and other methods," in *Genetic Algorithms+ Data Structures= Evolution Programs*, pp. 159–177, Springer, 1996.

[82] Wikipedia contributors, "Mutation (genetic algorithm) — Wikipedia, the free encyclopedia." `https://en.wikipedia.org/w/index.php?title=Mutation_(genetic_algorithm)&oldid=864910411`, 2018. [Online; accessed 1-November-2018].

[83] S. Sivanandam and S. Deepa, "Genetic algorithm optimization problems," in *Introduction to Genetic Algorithms*, pp. 165–209, Springer, 2008.

[84] B. H. F. Hasan and M. S. M. Saleh, "Evaluating the effectiveness of mutation operators on the behavior of genetic algorithms applied to non-deterministic polynomial problems," *Informatica*, vol. 35, no. 4, 2011.

[85] S. M. Lim, A. B. M. Sultan, M. N. Sulaiman, A. Mustapha, and K. Leong, "Crossover and mutation operators of genetic algorithms," *International Journal of Machine Learning and Computing*, vol. 7, no. 1, pp. 9–12, 2017.

[86] C. M. Mueller, M. Kaschub, C. Blankenhorn, and S. Wanke, "A cell outage detection algorithm using neighbor cell list reports," in *International Workshop on Self-Organizing Systems*, pp. 218–229, Springer, 2008.

[87] W. Xue, M. Peng, Y. Ma, and H. Zhang, "Classification-based approach for cell outage detection in self-healing heterogeneous networks," in *Wireless Communications and Networking Conference (WCNC), 2014 IEEE*, pp. 2822–2826, IEEE, 2014.

[88] T. Zhang, L. Feng, P. Yu, S. Guo, W. Li, and X. Qiu, "A handover statistics based approach for cell outage detection in self-organized heterogeneous networks," in *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*, pp. 628–631, IEEE, 2017.

[89] A. Zoha, A. Saeed, A. Imran, M. A. Imran, and A. Abu-Dayya, "A son solution for sleeping cell detection using low-dimensional embedding of mdt measurements," in *Personal, Indoor, and Mobile Radio Communication (PIMRC), 2014 IEEE 25th Annual International Symposium on*, pp. 1626–1630, IEEE, 2014.

[90] O. Onireti, A. Zoha, J. Moysen, A. Imran, L. Giupponi, M. A. Imran, and A. Abu-Dayya, "A cell outage management framework for dense heterogeneous networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 4, pp. 2097–2113, 2016.

[91] F. Chernogorov, J. Turkka, T. Ristaniemi, and A. Averbuch, "Detection of sleeping cells in lte networks using diffusion maps," in *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, pp. 1–5, IEEE, 2011.

[92] Y. Ma, J. Zhu, L. Liu, F. Lv, and Y. Wang, "A dynamic affinity propagation clustering algorithm based on mdt in self-healing heterogeneous networks," in *Communications and Information Technologies (ISCIT), 2016 16th International Symposium on*, pp. 318–323, IEEE, 2016.

[93] W. Feng, Y. Teng, Y. Man, and M. Song, "Cell outage detection based on improved bp neural network in lte system," 2015.

[94] Z. Jiang, Y. Peng, Y. Su, W. Li, and X. Qiu, "A cell outage compensation scheme based on immune algorithm in lte networks," in *Network Operations and Management Symposium (APNOMS), 2013 15th Asia-Pacific*, pp. 1–6, IEEE, 2013.

[95] M. Amirijoo, L. Jorguseski, R. Litjens, and L.-C. Schmelz, "Cell outage compensation in lte networks: algorithms and performance assessment," in *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, pp. 1–5, IEEE, 2011.

[96] F. Li, X. Qiu, W. Li, and H.-L. Wan, "High load cell outage compensation method in td-scdma wireless access network," *Journal of Beijing University of Posts and Telecommunications*, vol. 35, no. 1, pp. 32–35, 2012.

[97] A. Asghar, H. Farooq, and A. Imran, "Self-healing in emerging cellular networks: Review, challenges and research directions," *IEEE Communications Surveys & Tutorials*, 2018.

[98] A. Zoha, A. Saeed, A. Imran, M. A. Imran, and A. Abu-Dayya, "A learning-based approach for autonomous outage detection and coverage optimization," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 3, pp. 439–450, 2016.

[99] M. M. Al-Zawi, A. Hussain, D. Al-Jumeily, and A. Taleb-Bendiab, "Using adaptive neural networks in self-healing systems," in *Developments in eSystems Engineering (DESE), 2009 Second International Conference on*, pp. 227–232, IEEE, 2009.

[100] L. Xia, W. Li, H. Zhang, and Z. Wang, "A cell outage compensation mechanism in self-organizing ran," in *Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on*, pp. 1–4, IEEE, 2011.

[101] C.-Y. Chang, H.-J. Wang, H.-C. Chao, and J. H. Park, "Ieee 802.11 handoff latency improvement using fuzzy logic," *Wireless Communications and Mobile Computing*, vol. 8, no. 9, pp. 1201–1213, 2008.

[102] S. Mehta, N. Ullah, M. H. Kabir, M. N. Sultana, and K. S. Kwak, "A case study of networks simulation tools for wireless networks," in *2009 Third Asia International Conference on Modelling & Simulation*, pp. 661–666, IEEE, 2009.

[103] J. B. Ernst, S. C. Kremer, and J. J. Rodrigues, "A wi-fi simulation model which supports channel scanning across multiple non-overlapping channels in ns3," in *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, pp. 268–275, IEEE, 2014.

[104] B. Hu, "Analysis and modelling of wifi access point propagation data," *Mathematic Dept, University of York, UK*, 2011.

[105] S. M. Bilalb, M. Othmana, *et al.*, "A performance comparison of network simulators for wireless networks," *arXiv preprint arXiv:1307.4129*, 2013.

[106] T. Arvind, "A comparative study of various network simulation tools," *Int. J. of Comput. Sci. and Eng. Technol.(IJCSET)*, vol. 7, no. 8, pp. 374–378, 2016.

[107] E. Weingartner, H. Vom Lehn, and K. Wehrle, "A performance comparison of recent network simulators," in *2009 IEEE International Conference on Communications*, pp. 1–5, IEEE, 2009.

[108] NS-3, "ns-3 official website." `http://www.nsnam.org`, 2019.

[109] K. Wehrle, M. Günes, and J. Gross, *Modeling and tools for network simulation*. Springer Science & Business Media, 2010.

[110] A. Al-Ali and K. Chowdhury, "Simulating dynamic spectrum access using ns-3 for wireless networks in smart environments," in *2014 Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking Workshops (SECON Workshops)*, pp. 28–33, IEEE, 2014.

[111] K. Tan, D. Wu, A. J. Chan, and P. Mohapatra, "Comparing simulation tools and experimental testbeds for wireless mesh networks," *Pervasive and Mobile Computing*, vol. 7, no. 4, pp. 434–448, 2011.

[112] NS-3, "Propagation loss models ns-3." `https://www.nsnam.org/docs/models/html/propagation.html`, 2019.

[113] L. Maygua-Marcillo, L. Urquiza-Aguiar, and M. Paredes-Paredes, "Wireless channel 802.11 in ns-3," 2018.

[114] S. L. Cebula III, A. Ahmad, J. M. Graham, C. V. Hinds, L. A. Wahsheh, A. T. Williams, and S. J. DeLoatch, "Empirical channel model for 2.4 ghz ieee 802.11 wlan," in *Proceedings of the International Conference on Wireless Networks (ICWN)*, p. 1, Citeseer, 2011.

[115] S. Jung, C.-o. Lee, and D. Han, "Wi-fi fingerprint-based approaches following log-distance path loss model for indoor positioning," in *2011 IEEE MTT-S International Microwave Workshop Series on Intelligent Radio for Future Personal Terminals*, pp. 1–2, IEEE, 2011.

[116] T. K. Sarkar, Z. Ji, K. Kim, A. Medouri, and M. Salazar-Palma, "A survey of various propagation models for mobile communication," *IEEE Antennas and propagation Magazine*, vol. 45, no. 3, pp. 51–82, 2003.

[117] I. Stojmenovic, "Simulations in wireless sensor and ad hoc networks: matching and advancing models, metrics, and solutions," *IEEE Communications Magazine*, vol. 46, no. 12, pp. 102–107, 2008.

[118] U. Faigle and G. Turán, "Sorting and recognition problems for ordered sets," *SIAM Journal on Computing*, vol. 17, no. 1, pp. 100–113, 1988.

[119] T. Wang, G. Xing, M. Li, and W. Jia, "Efficient wifi deployment algorithms based on realistic mobility characteristics," in *The 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2010)*, pp. 422–431, IEEE, 2010.

[120] V. M. Nguyen and H. Claussen, "Efficient self-optimization of neighbour cell lists in macrocellular networks," in *21st Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1923–1928, IEEE, 2010.

[121] I. de-la Bandera, R. Barco, P. Munoz, and I. Serrano, "Cell outage detection based on handover statistics," *IEEE Communications Letters*, vol. 19, no. 7, pp. 1189–1192, 2015.

[122] K. Shi, Z. Ma, R. Zhang, W. Hu, and H. Chen, "Support vector regression based indoor location in ieee 802.11 environments," *Mobile Information Systems*, vol. 2015, 2015.

[123] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in *Advances in neural information processing systems*, pp. 155–161, 1997.

[124] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[125] U. Mehboob, J. Qadir, S. Ali, and A. Vasilakos, "Genetic algorithms in wireless networking: techniques, applications, and issues," *Soft Computing*, vol. 20, no. 6, pp. 2467–2501, 2016.

[126] D. Devaraj and B. Yegnanarayana, "Genetic-algorithm-based optimal power flow for security enhancement," *IEE Proceedings-*

Generation, Transmission and Distribution, vol. 152, no. 6, pp. 899–905, 2005.

[127] N. M. Razali, J. Geraghty, *et al.*, "Genetic algorithm performance with different selection strategies in solving tsp," in *Proceedings of the world congress on engineering*, vol. 2, pp. 1134–1139, International Association of Engineers Hong Kong, 2011.

[128] S. L. Yadav and A. Sohal, "Comparative study of different selection techniques in genetic algorithm," *Journal Homepage: http://www. ijesm. co. in*, vol. 6, no. 3, 2017.

[129] N. R. Kishor, "International journal of advance research in computer science and management studies," *International Journal*, vol. 2, no. 3, 2014.

[130] T. Blickle and L. Thiele, "A comparison of selection schemes used in genetic algorithms," 1995.

[131] A. Umbarkar and P. Sheth, "Crossover operators in genetic algorithms: A review.," *ICTACT journal on soft computing*, vol. 6, no. 1, 2015.

[132] L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval-schemata," in *Foundations of genetic algorithms*, vol. 2, pp. 187–202, Elsevier, 1993.

[133] J. F. Gonçalves, J. J. de Magalhães Mendes, and M. G. Resende, "A hybrid genetic algorithm for the job shop scheduling problem," *European journal of operational research*, vol. 167, no. 1, pp. 77–95, 2005.

[134] S. Sivanandam and S. Deepa, *Introduction to genetic algorithms*. Springer Science & Business Media, 2007.

[135] A. Sorsa, R. Peltokangas, and K. Leiviska, “Real-coded genetic algorithms and nonlinear parameter identification,” in *2008 4th International IEEE Conference Intelligent Systems*, vol. 2, pp. 10–42, IEEE, 2008.

[136] L. Hao, B. Ng, and Y. Qu, “Dynamic optimization of neighbor list to reduce changeover latency for wi-fi networks,” in *Proceedings of the 2017 International Conference on Telecommunications and Communication Engineering*, pp. 20–24, ACM, 2017.

[137] Y. Lee, K. Kim, and Y. Choi, “Optimization of ap placement and channel assignment in wireless lans,” in *27th Annual IEEE Conference on Local Computer Networks, 2002. Proceedings. LCN 2002.*, pp. 831–836, IEEE, 2002.

[138] E. L. Fernandes and C. E. Rothenberg, “OpenFlow 1.3 software switch,” *Salao de Ferramentas do XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos SBRC*, pp. 1021–1028, 2014.

[139] “Ieee standard: Radio resource measurement of wireless lans,” *IEEE Std 802.11k-2008 (Amendment to IEEE Std 802.11-2007)*, pp. 1–244, June 2008.

[140] P. Yu, F. Zhou, T. Zhang, W. Li, L. Feng, and X. Qiu, “Self-organized cell outage detection architecture and approach for 5g h-cran,” *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[141] C. J. Burges, “A tutorial on support vector machines for pattern recognition,” *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[142] S. Hu, Y.-d. Yao, and Z. Yang, “Mac protocol identification using support vector machines for cognitive radio networks,” *IEEE Wireless Communications*, vol. 21, no. 1, pp. 52–60, 2014.

[143] J. Choi, H. Kim, C. Choi, and P. Kim, "Efficient malicious code detection using n-gram analysis and svm," in *Network-Based Information Systems (NBiS), 2011 14th International Conference on*, pp. 618–621, IEEE, 2011.

[144] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[145] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[146] N. Pater, "Enhancing random forest implementation in weka," in *Machine learning conference paper for ECE591Q*, 2005.

[147] S. Ren, X. Cao, Y. Wei, and J. Sun, "Global refinement of random forest," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 723–730, 2015.

[148] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[149] S. Tan, "An effective refinement strategy for knn text classifier," *Expert Systems with Applications*, vol. 30, no. 2, pp. 290–298, 2006.

[150] K.-j. Kim and H. Ahn, "A recommender system using ga k-means clustering in an online shopping market," *Expert systems with applications*, vol. 34, no. 2, pp. 1200–1209, 2008.

[151] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," *Computers & security*, vol. 21, no. 5, pp. 439–448, 2002.

[152] H. Raeisi Shahraki, S. Pourahmad, and N. Zare, "Important neighbors: A novel approach to binary classification in high dimensional data," *BioMed research international*, vol. 2017, 2017.

[153] Z. Deng, X. Zhu, D. Cheng, M. Zong, and S. Zhang, "Efficient knn classification algorithm for big data," *Neurocomputing*, vol. 195, pp. 143–148, 2016.

[154] J. W. Branch, C. Giannella, B. Szymanski, R. Wolff, and H. Kargupta, "In-network outlier detection in wireless sensor networks," *Knowledge and information systems*, vol. 34, no. 1, pp. 23–54, 2013.

[155] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, "A comparative study of anomaly detection schemes in network intrusion detection," in *Proceedings of the 2003 SIAM International Conference on Data Mining*, pp. 25–36, SIAM, 2003.

[156] Y. El Mourabit, A. Bouirden, A. Toumanari, and N. Moussaid, "Intrusion detection techniques in wireless sensor network using data mining algorithms: comparative evaluation based on attacks detection," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 6, no. 9, pp. 164–172, 2015.

[157] X. Yang, Y. Wang, D. Wu, and A. Ma, "K-means based clustering on mobile usage for social network analysis purpose," in *2010 6th International Conference on Advanced Information Management and Service (IMS)*, pp. 223–228, IEEE, 2010.

[158] J. Zhang, "Advancements of outlier detection: A survey," *ICST Transactions on Scalable Information Systems*, vol. 13, no. 1, pp. 1–26, 2013.

[159] K. Zhang, G. Chuai, W. Gao, X. Liu, and Y. Ren, "Data analysis of measurement report and diagnosis of mobile network malfunction

based on k-means algorithm,” in *International Conference in Communications, Signal Processing, and Systems*, pp. 170–181, Springer, 2017.