Self-Adaptive WLAN Access Point for Optimizing Network Performance Using Multi-Objective Genetic Algorithm (MOGA)

Joel C. Delos Angeles and Elmer P. Dadios

***Abstract* — Current deployment of WLAN access points (AP) require manual configuration of wireless parameters. Wireless parameters are commonly set haphazardly without being aware of the basic wireless conditions. This paper proposes a self-adaptive AP based on genetic algorithms (GA). The AP adapts to interference and link quality of client stations. Interference is mitigated and client link quality is improved or optimized. A chromosome consists of genes of parameters such as frequency channel, channel width, maximum data rate, maximum transmit power, and guard interval. Often competing objectives such as mitigating interference, maximizing the data rate, and minimizing the error rates necessitate that the GA be multi-objective. The MOGA comes up with the fittest candidates by running them through a fitness function which scores the genes based on the survey scan of other interferer AP and the wireless performance statistics of client devices. The GA's chosen configuration is applied and its effect is continuously assessed. Finally, the result of the self-adaptive WLAN AP genetic algorithm is compared against the Linux hostapd Automatic Channel Selection scheme.**

***Index Terms*— access point, AP, IEEE 802.11n, WiFi, automatic configuration, interference, self-adaptive, genetic algorithm, GA, multi-objective, MOGA**

I. Introduction

WIRELESS devices have become so ubiquitous that there are now almost as many mobile phones as there are people in the world. Most people also own at least two mobile phones from different mobile operators for voice, short messaging service, and Internet use. Moreover, demand for Internet speed and reasonable cost of access resulted in mobile phones having multiple radio access technologies – such as mobile 2G/3G/4G and WiFi. The choice of which radio access technology to use when connecting to the Internet is driven by quality, cost, and availability resulting to what is called Heterogenous Networks (HetNets). In HetNets, access to the Internet does not converge to a single wireless access technology but instead takes advantage of the different wireless technologies available to the user. Users can even share their Internet access with other mobile phone users within their vicinity using hotspot tethering. In hotspot tethering, multiple devices connect, using WiFi or Bluetooth, to a primary device which accesses the Internet access via 3G or 4G. Direct wireless connection to a high-power base station tower is replaced with ad-hoc user-to-user connectivity and the primary device is said to act as a WiFi base station. Eventually, this trend could result to a new cellular paradigm shift where there will be more base stations than cellular phones [1].

Frequency Channel	Channel Setting	Max Data Rate	Max Transmit Power Reduction	Guard Interval
1 to 11 (or 13 in some AP)	HT20 or HT40+ or HT40-	MCS0 to 7 (or with 8 to 14 for HT 40 MHz)	in dB (to be subtracted from max tx power of AP)	400 or 800 nsec

Fig 1. Chromosome representation and example showing the wireless parameters available for manipulation by the GA. Note that transmit power reduction (in dB) is used for the gene instead of absolute unit of maximum power (in dBm) which varies from AP to AP

As the number of hotspots increase, new ways of managing spectrum use and interference are called for. Traditional methods of interference management like frequency reuse or base station coordination do not directly translate to HetNets [1]. Current hotspot

Joel C. Delos Angeles is affiliated with Electronics and Communications Engineering, De La Salle University - Dasmarinas, Dasmarinas City, Cavite, Philippines (e-mail: joel_delosangeles@dlsu.edu.ph)

Elmer P. Dadios is affilaited with Manufacturing Engineering and Management, De La Salle University - Manila, Manila, Philippines (e-mail: elmer.dadios@dlsu.edu.ph)

access point (AP) deployments are uncoordinated, particularly the choice of WiFi frequency channel in commercial establishments, residences, or personal tethered hotspots. One can easily find that 2.4 GHz channel frequencies overlap by performing a spectrum scan using a laptop with software like inSSIDER from metageek.com. Of course, non-overlapping 20 MHz frequency channels can be used for 3 nearby access points – channels 1, 6, 11 standing for 2.412, 2.437, and 2.462 GHz respectively. This is an ideal and desired scenario but the deployment becomes more complex as the number of access points in the area grows, as in HetNets. Furthermore, the WiFi IEEE 802.11n standard implements 40 MHz channels to double the data rate making operating in overlapping channels more likely.

From an AP's perspective, any interferer operating or leaking into its own *frequency channel* of operation is a co-channel interferer. Co-channel interference is not the only type of interference that can affect an AP. An interferer present in an adjacent channel is an adjacent channel interferer. Adjacent Channel Interference (ACI) causes problems that are related to the carrier sensing mechanism in IEEE 802.11 and are especially severe in multi-radio systems, where the radios are very closely spaced [2]. The number of available orthogonal (or non-interfering) channels in 2.4 GHz IEEE 802.11n depends on the spatial spacing between the radios, the *channel width* (HT20 vs. HT40), and traffic pattern. In a multi-radio system scenario, the separation between the three non-overlapping WiFi channels is almost nullified. The situation becomes worse since no two frequency channels can be considered orthogonal. ACI can be addressed by placing enough spatial separation between the access points. If there are constraints to the space limitations, the only option to overcome ACI problems is through *transmit power* control [2]. For this study, *frequency channel*, *channel width*, and *transmit power* are some of the AP wireless parameters included in the chromosome for evaluation by the genetic algorithm (GA). Figure 1 show all the parameters used in this paper and includes *maximum data rate* and *guard interval (GI).* Maximum data rate puts a limit on the maximum modulation coding scheme (MCS) for the downlink (AP to STA). Guard interval by default is 800 nanoseconds although this can be adjusted to 400 nsec during better network conditions to increase network throughput but at the risk of higher transmission errors. In this research, only 802.11n radios are used but in no way does this constraint limit the validity of the findings to 802.11n only. The same theoretical wireless concepts and experimental results should apply to 802.11a/b/g or even to mobile technologies like 3G and 4G.

The chromosome's genes represent the adjustable wireless parameters in a given radio, and by genetically manipulating the chromosomes, the GA can find a set of parameters which optimize the radio to meet certain objectives. Some of these objectives can be utilized to improve performance and Quality of Service (QoS), to enhance spectrum usage in the midst of interferers, or to further advance wireless ubiquity [3].

A genetic algorithm that takes into consideration multiple and often-competing objectives for optimization and decision making is a multi-objective GA (MOGA). A paper written by Rondeau et al. [3], as they developed the GA-based adaptive component of a cognitive radio in Virginia Tech (VT) Center for Wireless Telecommunications (CWT), gave a thoughtful consideration on the application of a multi-objective genetic algorithm (MOGA) to a wireless system. Rondeau et al. cited the limitation of a particular GA selection and evaluation method in which evaluations along different dimensions are combined into a single metric. In the case of wireless communications, the dimensions can be bit error rate (BER), bandwidth, power consumption or network latency, to name a few. According to this paper, the single-metric method breaks down in cases where the values of the dimensions can vary greatly in magnitude (as in BER of 10^{-6} versus data rate of 10^{6}) and normalizing each dimension requires a great deal of domain knowledge. Nonetheless, one contribution of the present study is to propose a single metric, which will be called *PRR-MCS*, to evaluate and score each of the genes in a chromosome (Figure 1). *PRR-MCS* is the packet reception rate in percent (%) multiplied by the MCS data rate in megabits per second (Mbps). Each gene is given a score whose unit is in terms of PRR-MCS. As the wireless parameters or genes reflect the dimensions of the MOGA, normalization across the different dimensions is simplified since all of the gene scores have a common unit of measure. Basic operations such as summing, averaging, or weighted sum/average of the gene scores can then be used to operate an AP toward a desired objective such as optimizing network performance in the presence of radio interference.

Genetic algorithms have common processes such as the definition and representation of data into genes and chromosomes, the operations of crossover and mutation, the selection of chromosome for the succeeding generations, and the existence of a fitness function to determine chromosome fitness. A main contribution of the present study is to design and implement into a software code a genetic algorithm with a fitness function which utilizes two key inputs: (a) a survey scan of interferer access points operating in the area and (b) associated client station (STA) statistics such as signal level, packet retransmit and failure rates, and MCS data rates for uplink and downlink. The genetic algorithm used is multi-objective and takes into consideration certain dimensions such as frequency channel of operation, bandwidth, maximum data rate, and maximum transmit power. Finally, to the knowledge of the authors, the metric proposed to score each of the chromosome genes is a novelty. PRR-MCS is a common unit to evaluate each of the wireless parameters or genes which the self-adaptive AP takes in as recommended configurations.

In this paper, Section 2 cites related researches to the current subject. Section 3 covers the background of the problem and the objectives of this paper as well as its scope. Section 4 describes the MOGA as a method to converge to a solution to the problem. The fitness function is discussed in detail in Section 5, specifically how each of the chromosome genes is scored and how PRR-MCS for each is computed. Section 6 discusses the experimental result and also gives a comparison of the proposed self-adaptive GA and the Automatic Channel Selection (ACS) feature of `hostapd`. Finally, Section 7 concludes the present paper and identifies future work and enhancements.

II. Related Work

The need to automate the deployment and configuration of radio access points or base stations to achieve certain objectives is extensively studied [2],[4],[5]. The studies can be classified in general as (1) whether the optimization applies to a whole network planning and deployment, or 2) whether the configuration applies locally to a single radio or access point. As will be shown in the following discussion, while there are numerous literatures related to the former subject, the latter subject seems to lack significant attention. The present paper falls under the latter category as it proposes self-adaptation of a single AP to interfered states and optimizing network performance using the wireless statistics of associated client stations.

For network-wide application of wireless parameter configurations, Zubow, et al. [2] recommend sufficient spatial spacing to obtain more orthogonal channels out of the 2.4 GHz WiFi band. Moreover, control of transmit power becomes vital in space-limited, multi-radio systems. Garcia-Saavedra, et al. [4] presents a novel Self-Optimizing, Legacy-Compatible Opportunistic Relaying (SOLOR) framework, which optimizes the network topology and relay schedules while considering different node performance and power consumption trade-off preferences. Recommendations in optimizing the configuration of wireless parameters have been investigated, often targeting automatic adaptation to specific trade-off preferences. A load-balancing algorithm for reducing Radio Frequency (RF) Electromagnetic Fields (EMF) exposure while maintaining a level of QoS performance has also been proposed by Sidi, et al. [5]. They developed a stochastic approximation based self-optimizing algorithm that dynamically adapts the network to reduce the exposure index (EI) in a heterogeneous network with macro- and small cells.

Some works on radio network planning and deployment used genetic algorithms to optimize the radio configurations. Such works investigated the usability of genetic algorithms for optimizing wireless mesh networks. Pries, et al focused on the routing and channel assignment in large-scale wireless mesh networks to achieve a max-min fair throughput allocation [6]. It should be noted that the terms mesh and relay have something in common with user-to-user type connectivity such as hotspot tethering. A few other papers focused channel assignment problems and planning [7] - [11]. The paper by Chia, et al. [7] introduces an adaptive genetic algorithm (GA)-based channel assignment strategy for resource management and to reduce the effect of EMC interferences. Fu, et al. [8] developed a new heuristic algorithm which includes GA to tackle the same channel assignment problem to assign a minimum number of channels under certain constraints to requested calls in a cellular radio system. Ding et al. [9] used a weighted conflict graph to model interference between wireless links more accurately. They also presented a novel genetic algorithm to demonstrate that the network performance can be dramatically improved by properly utilizing partially overlapping channels. The genetic algorithm was also found to outperform the greedy algorithm in mitigating the interference within the network leading to higher network throughput. The paper by Jalili, et al. [10] compared Taboo Search and GA in planning and optimization of 3rd Generation (3G) Universal Mobile Telecommunication System (UMTS) networks. Finally, Cacciani, et al. [11] attempted to solve the problem of identifying sites where to place the resources (or access points) for the optimal coverage of a given area using genetic algorithms.

Genetic algorithms continue to find more applications in mobile wireless networks. The paper by Paikaray [12] presents a design of an adaptive multi-attribute, vertical handoff decision algorithm for 4th Generation mobile networks based on fuzzy logic and genetic algorithms. The minimization of the number of handoffs in heterogenous 4G networks has been shown to be achievable in the paper by Chandralekha and Behera [13] through optimization of network parameter values. They proposed a multi criteria vertical handoff decision algorithm which will select the best available network with optimized parameter values (such as cost of network should be at a minimum). The decision problem was formulated as multiple objective optimization problems and simulated using genetic algorithm.

In comparison to studies of network-wide optimization of wireless parameters, papers which aim to develop GA-based programs for a single access point are limited. One such study uses GA for cognitive radios [3]. This paper by Rondeau et al. is part of an initiative by Virginia Tech (VT) Center for Wireless Telecommunications (CWT) to develop a cognitive radio engine and presents its adaptive component which uses GA and is cited quite extensively in the present study.

A few more references were used for the present paper as they provide key concepts and data for the computations used in the program code. Zhang, et al. [14] emphasized a limitation of WiFi IEEE 802.11 protocols in handling frame losses which are not due to link quality but is rather due to interference. In their study, rate adaptation is guided by signal-to-noise ratio (SNR) for handling of interference. It should be noted that rate adaptation is just one of the components of the self-adaptive access point. As shown in the chromosome genes, frequency channel selection and transmit power optimization are also features of the self-adaptive AP. Like the present study and the paper by J. Zhang et al., another paper which attempted to restore confidence to theoretical prediction of wireless link quality is that of D. Halperin et al. [15]. They introduced the concept of effective SNR to make packet delivery predictions. As for the guard interval parameter, the literature is limited and the effect of changing the GI from the default value is not well-studied. Measurements conducted by the University of Hampshire Interoperability Laboratory [17] estimated the effect of GI on packet error rates. Experimental result of software program coded in C++ for the self-adaptive AP is compared to that of the Automatic Channel Selection (ACS) feature of Linux-based access points.

`Hostapd` is a user-space daemon commonly used in Linux-based access points. One of its options is Automatic Channel Selection (ACS) so that a WiFi device can automatically figure out which channel to operate on depending on the level of interference. ACS utilizes the same information provided by `iw wlan0 survey dump`. ACS introduces a metric called *interference factor* which is computed from the formula: (busy time – tx time) / (active time – tx time). The rationale for this formula is given in [16]. The formula is intuitive in that it gives the percentage of time in which a channel is busy. This ratio is also used in the current study. However, the ratio is not treated as a final metric, rather the ratio is used to give a correction factor to the interference noise floor generated from survey scan of interferer access points. The correction factor was found reasonable as in the case where an AP does not generate any data traffic and just transmits beacon signals. The average transmit power is approximated as the signal power during beacon transmissions multiplied by the channel busy time divided by the channel active time. The correction of the interference noise floor using the busy time / active time ratio is discussed in detail in the previous sections.

The ACS first takes the average of 5 readings of the busy time/active time ratio. This average becomes the interference factor for a channel. It does this for all the channels on which an AP can operate on (channels 1 up to 11 or up to 13). When the 11 readings are computed, ACS computes the total interference for each channel. It does this by summing up the interference factor for 5 neighboring channels since 5 channels always overlap with a specific channel of choice. If HT40+ or HT40-

is used, ACS will add up the interference factors of 9 neighboring channels. The 5 or 9 neighboring channels apply for the centermost channels and not the ones on the edges. For the edge channels 1 and 11, for instance, only 3 channels will be summed up while for channels 2 and 10, 4 channels will be summed up. A similar scheme will apply to 40 MHz configurations.

With the current computation of ACS of simply summing up the interference factors, it is obvious that a 40 MHz channel will never be chosen by the algorithm since it will always have a higher sum of interference factors than that of 20 MHz channel. Thus, the self-adaptive AP algorithm will be compared with a modified ACS algorithm in which the total interference is taken as the average, not the sum, of interference factors of 5 (for 20 MHz) or 9 (for 40 MHz) neighboring channels. Also, instead of taking 5 readings for each channel, only 2 readings will be performed which are spaced 5 seconds apart and the busy time/active time ratio becomes the ratio of the difference of two busy times to the difference between two successive active time readings.

Unlike the self-adaptive AP GA algorithm, ACS does not take into account how much a channel is affecting the channel of operation. It is expected that the effect of one channel is less as its distance from the channel of choice increases. Thus, the authors do not see enough justification to simply average the interference factors. A weighted average could be more fit in quantifying the interference to a channel. It can also be observed that the interference in ACS is quantified as ratio of busy and active times. The self-adaptive AP expresses interference in the proper unit – dBm or mWatts. Thus, although ACS can tell what percentage of time a channel is busy, it says nothing about how strong is the power of interferer signal making the channel busy. The self-adaptive AP starts with the strength of the interferer access points to generate an interference noise floor for the whole 2.4 GHz band. After that, this interference noise is corrected because the signal strengths measured are not present in the air for the whole time. The busy time/active time ratios for several channels are gathered for the whole band, these ratios are expressed in dB, and they are subtracted from the initial interference noise floor. A major advantage of the self-adaptive AP algorithm is the use of correct units in expressing interference. Nevertheless, the ACS has the properties of simplicity of computation and ease of implementation. Now that the ACS has been introduced and compared with the self-adaptive AP algorithm, this paper will proceed with the quantitative comparison between the results of both algorithms.

III. Background and Scope of the Study

Rondeau et al. gave 3 parts of a cognitive radio which made it cognitive: the ability to sense the RF spectrum even at a minimum sensing, geographical surroundings, and the user's needs; the capacity to learn, ideally in both supervised and unsupervised modes; and finally, the capability to adapt within any layer of the radio communication system [3]. This definition is useful in scoping the current paper.

The primary objectives of a self-adaptive Wireless LAN (WLAN) AP are: 1) to operate optimally in a wireless medium in which interferers are present and 2) to do so while improving its wireless performance to deliver sufficient QoS to its connected clients. As in some of the citations in the related works, it may also be desirable to have a secondary objective where human exposure to EMF is reduced after attaining the primary objectives. Thus, in the context of the definition of cognitive radios by Rondeau, et al., the self-adaptive AP can do the following: sense the RF spectrum to detect interferers and to gather wireless statistics of associated client stations performance. It can also learn which configurations are most fit using a genetic algorithm based on the knowledge of other interfering access points and the wireless performance of the client devices.

The scope of the present study is self-adaptation *only* at the level of the wireless configuration which is accessible to users and administrators. The self-adaptive AP automates the configuration by network administrators and even improves on it by making informed decisions on which combination of values of wireless configuration parameters to use. In addition to this, the adjustments made by the self-adaptive AP to its own configurations are dynamic due to changing wireless network conditions. For instance, the turning on of a new interferer AP will result to a new configuration evaluated by the genetic algorithm. The self-adaptive AP is informed of the new recommended configuration and applies it at appropriate times to reduce instability and minimize the downtime of associated clients. The wireless performance of associated clients is also taken into consideration in the decision to recommend new wireless configuration. Note that this adaptation mechanism is outside of the existing MAC layer mechanisms in IEEE 802.11 such as data rate adaptation. Adaptation at the 1PHY or MAC layers and protocols is outside the scope of this study. Moreover, most previous works on rate adaptation rely only on frame losses to infer channel quality, but performs poorly if frame losses are mainly caused by interference [14].

IV. The Self-Adaptive AP Program Flow

This section will discuss the overall flow of the program whose code will be installed on a Linux-based Access Point (AP) to make it self-adaptive and optimize itself for network performance under interfered scenarios. Another key topic of this section is the fitness function, which utilizes a single-metric (PRR-MCS) method and normalizes each dimension of the multi-objective genetic algorithm (MOGA). The fitness function uses the PRR-MCS to evaluate and score each of the genes in a chromosome (Figure 1).

This section also identifies the commands and programs used to gather the data needed by the fitness function. These commands or programs are installed in a Linux-based access point in order to perform data gathering functions such as channel survey scan and client station performance statistics. Whenever mentioned, these commands or programs are formatted in the `Courier font.` The device interface under test is also represented as `wlan0` to facilitate understanding of the command usage.

Before going through the details and operation of the fitness function used by the GA, it is worth mentioning what operations are inside the multi-objective GA cycle in Figure 2. The internals of the MOGA is shown in Figure 3. First, the genes of the initial chromosomes are randomly generated. The population size is 50 chromosomes whose format is shown in Figure 1. A function in the C++ code, called objective_channel(), generates the interference noise floor for the whole 2.4 GHz band from 2400 – 2483 MHz. The fitness of the chromosomes are evaluated using the output of the objective_channel() function and the chromosomes are given their corresponding

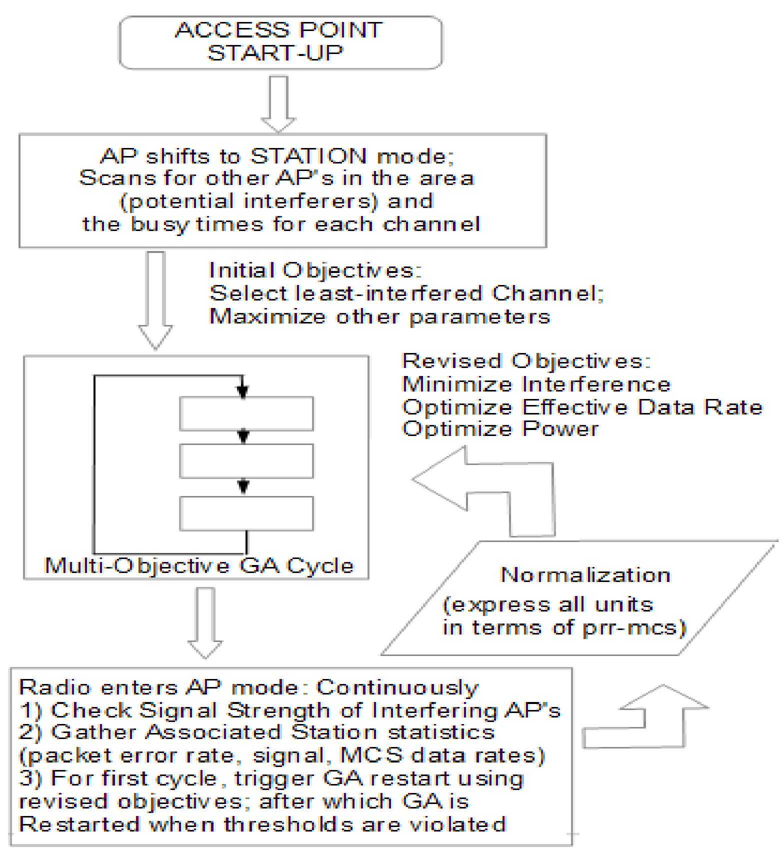

Fig. 2. Over-all flow of the program whose code is installed in an Access Point (AP) to make it self-adaptive

scores. A penalty function is also used to remove disallowed configurations resulting from incorrect combinations of channel frequency and channel setting. The chromosomes are sorted from the highest score to the lowest score for easier processing. After sorting, the standard GA operations are then applied to the initial population. The crossover() function performs single point crossover between neighboring chromosomes (parents) in the sorted array at a crossover probability of 90%. This operation produces a new generation of offspring. The mutation() function introduces random changes to the genes at a mutation probability of 20%. Fitness is again evaluated for the new generation of offspring. Then, the selection() function selects the top 50 chromosomes from the set of 100 composed of the previous generation of parents and the current generation of offspring. This cycle is repeated for 5 to 10 generations to converge to a solution. In Figure 2, this initial run of the GA cycle seeks to find a solution with the objective of identification of a least-interfered channel while maximizing the other parameters of transmit power and MCS data rate (GI is also at default of 800 nsec). In this run of the GA, the fitness function relied on the objective_channel() function which uses Linux commands to survey the interferer access points in the air. These commands are discussed in detail below. Another function, the objective_station_dump() function, which uses the wireless performance statistics of client stations associated to the AP, will be used together with objective_channel() for the succeeding runs of the GA cycle. The objectives are also revised

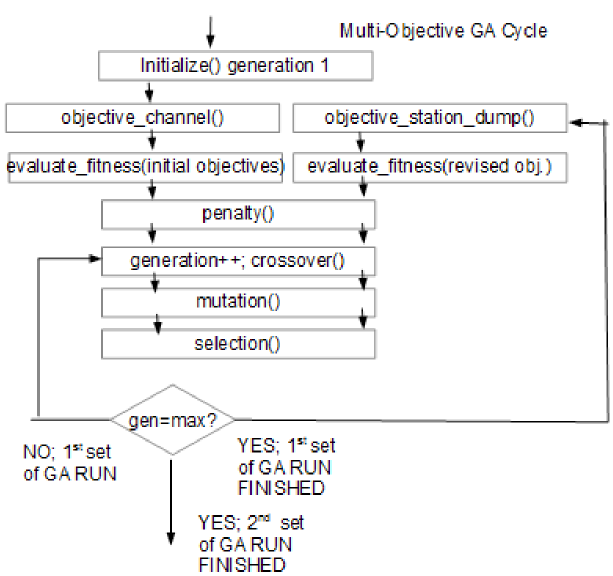

Fig. 3. The internals of the MOGA Cycle shown as a block in Figure 2

when the fitness function is called. Interference mitigation is still part of the revised objectives. For the succeeding runs of the GA cycle, however, the revised objectives include optimization of transmit power and MCS data rate instead of just maximization of these parameters. The standard operations of genetic algorithms are also shown in Figure 3.

V. The Fitness Function

Upon start-up of the Linux-based device, it operates as a client station (STA) instead of an AP. This is necessary in order to gather a complete survey scan of the other interferer access points operating in an area. Although this feature is also possible while the device is operating as an AP using `iwinfo wlan0 scan`, a more complete set of data is gathered using `iwlist wlan0 scanning`. The latter command shows the secondary channel which an interferer AP uses and this piece of information is not in the former command. An AP uses a secondary channel when it operates in HT40+ or HT40- channel setting which means that the bandwidth used is 40 MHz instead of the usual 20 MHz. Figure 4 illustrates a sample of the data gathered by `iwlist wlan0 scanning` and what the program will construct out of these raw data.

```
Cell 01 - Address: xx:xx:xx:xx:xx:xx
...
          Mode: Master  Channel: 1
          Signal: -56 dBm  Quality: 54/70
...
Cell 02 - Address: xx:xx:xx:xx:xx:xx
...
          Frequency:2.462 GHz (Channel 11)
          Quality=53/70  Signal level=-57 dBm
...
  HT operation:
       * primary channel: 11
       * secondary channel offset: below
```

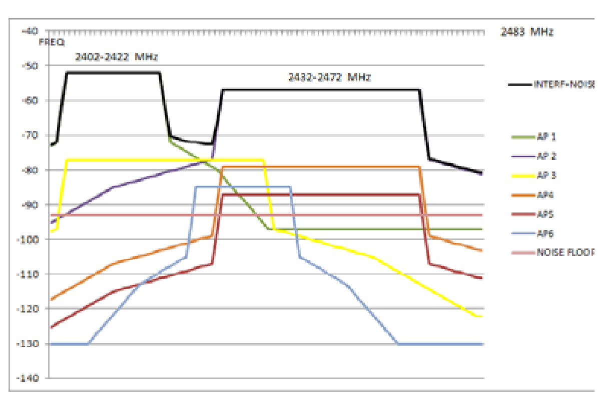

Fig 4. Sample iwlist wlan0 scanning output (truncated … and MAC addresses replaced with xx:xx:xx:xx:xx:xx) and the equivalent interference noise floor

The self-adaptive AP program makes use of the output of `iwlist wlan0 scanning` to capture the interference caused by all the other access points currently present in the air medium. Then, it constructs an interference noise floor which is the sum of the noise floor (around -93 dBm) and the contribution of each AP to its specific channel of operation. The net result if the t opmost line in the graph in Figure 4. A few more items need further explanation in the figure. If the secondary channel offset is present in the output of the program, it means that the channel used by that AP is 40 MHz instead of the usual 20 MHz. Furthermore, an above offset means that the AP occupies its primary channel and the next channel (e.g. 1+5) while a below offset means that the primary channel and channel below are used (e.g. 11+7).

The access point is also operated as a station (STA) initially to gain access to another crucial command - `iw wlan0 survey dump`. This command measures the amount of time a chosen channel is busy and the channel active time. Just before running this command, it is necessary to place the STA at the chosen for scanning through `iw wlan0 set channel <number>`. For this study, the following WiFi channels are scanned: 1 (2402-2422 MHz), 5 (2422-2442 MHz), 9 (2442-2462 MHz), and 11 (2452-2472 MHz). The readings at channels 1, 5, and 9 are used as is while the reading at channel 11 is divided by 2 to compensate the fact that the channel 9 scan already covered half of channel 11.

It will be shown later that the ratio of the busy time and the active time is a good correction factor to the interference noise floor (Figure 4) computed from the signal levels (in dBm) of all the interferer access points. If the computed interference noise floor used remains unchanged in the program, the level of interference is overestimated. The signal level received from interferer access points are only those of beacon transmissions which are typically broadcasted by an AP every 100 milliseconds and occupying the medium only for a brief 50 bytes at 1 Mbps data rate. After computing the interference noise floor, the busy time/active time ratio for the chosen channels are expressed in dB and are subtracted from the interference noise floor for those channels resulting to reduction to more reasonable values. Figure 5 shows a sample output of `iw wlan0 survey dump` and the corrected interference noise floor. As already mentioned, channels 1, 5, 9, and 11 are measured for busy and active times. This measurement is done twice and the differences for each parameter are used to compute for the ratio. A more accurate formula which is also used by the `hostapd` Automatic Channel Selection (ACS) algorithm is (busy time - transmit time) ÷ (active time - transmit time) although this value is very near the busy time ÷ active time.

```
Survey data from wlan0
        frequency:  2412 MHz [in use]
        noise:      -95 dBm
        channel active time:  3466683 ms
        channel busy time:     386128 ms
        channel receive time:  306527 ms
        channel transmit time:  22693 ms
…
Survey data from wlan0
        frequency:  2412 MHz [in use]
        noise:      -95 dBm
        channel active time:  3467674 ms
        channel busy time:     386144 ms
        channel receive time:  306527 ms
        channel transmit time:  22703 ms
```

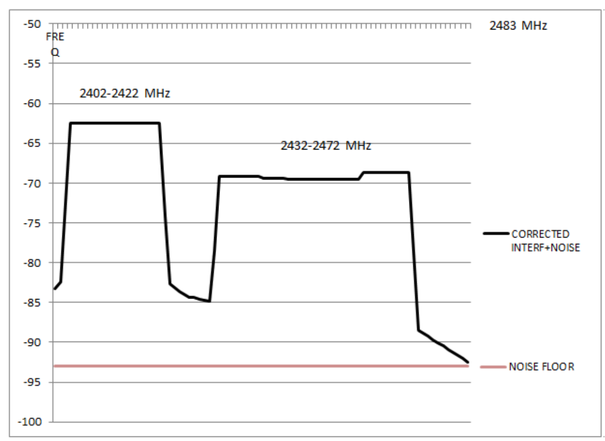

Fig. 5. Sample iw wlan0 survey output (truncated …) and the corrected interference + noise floor

In summary, the AP is started in STA mode in order to gain access to two Linux commands: `iwlist wlan0 scanning` and `iw wlan0 survey dump`. When the device enters AP mode which is its normal mode of operation, it is still useful for it to be alerted when new interferer access points are turned on. The command `iwinfo wlan0 scan here` becomes valuable. This command does not require the AP to go back to STA mode. Instead, the AP can regularly run this command in the background while serving STA clients accessing the network. The AP is effectively alerted if there are new interferer access points in the air. The AP can then defer a more thorough survey scan using the two main commands mentioned above at scheduled times, or when there are no connected client stations. So far, the results of the survey of interferer access point are enough for the GA to select a frequency channel (1 to 11 or 13 in some AP) and the channel setting (HT20, HT40- below, or HT40+ above). For the initial objective, power and MCS data rate will be maximized and the guard interval will stay at 800 nsec. After the recommendation from the GA, the AP configures itself accordingly and clients can now associate to it.

The fitness function utilizes two key inputs: (a) a survey scan of other access points operating in the area and (b) associated client station statistics such as signal level, packet retransmit and failure rates and MCS data rates for uplink and downlink. The first input has already been discussed in detail above. In the C++ code, the first input is used by the objective_channel() function. This leads the other input to the fitness function, the associated client station statistics, used by the objective_station_dump() function. With the device now operating in AP mode, the program calls another Linux command `iw wlan0 station dump`. Figure 6 shows a sample output for a single STA associated to the AP. The data from all the associated STA are gathered and will be used to further optimize the operation of the AP. Furthermore, for more significant readings, the AP generates `ping` traffic to all associated client STA before running the station dump. A `bash` script is written for this purpose and the main *C++* code uses a system() call to occasionally run this `bash` script.

```
Station xx:xx:xx:xx:xx:xx (on wlan0)
        inactive time:  40 ms
        rx bytes:       38189
        rx packets:     351
        tx bytes:       37705
        tx packets:     521
        tx retries:     339
        tx failed:      8
        signal:         -60 dBm
        signal avg:     -61 dBm
        tx bitrate:     19.5 MBit/s MCS 2
        rx bitrate:     6.5 MBit/s MCS 0
```

Fig. 6. Sample `iw wlan0 station dump` output for a single STA associated to the AP

```
STA RX_SIGNAL RX_MCS TX_MCS TX_RETRY TX_FAIL
XX:XX:XX:73:42:68 -53.62 39.00 19.50 0.39 0.01
XX:XX:XX:3b:0a:b4 -61.46 26.00 26.00 0.68 0.04
XX:XX:XX:8d:a7:0d -65.69  6.50 52.00 0.23 0.01
XX:XX:XX:89:30:e3 -49.23 58.50  6.50 0.18 0.00
```

Fig. 7. Sample summary of `iw wlan0 station dump` as generated by a `bash` script (xx:xx:xx hides the device manufacturer)

The single-metric of PRR-MCS which will be used to score each of the genes in a chromosome is the prominent feature of the objective_station_dump() function. Each gene score will be expressed in terms of PRR-MCS which is the packet reception rate % multiplied by a corresponding MCS data rate in Mbps. In order for the C++ code not to become littered with command parsing functions, the already mentioned `bash` script also summarizes the output of `iw wlan0 station`. Figure 7 shows a sample output of the bash script.

The statistics in Figure 7 need some explanation. STA is, of course, the MAC address of the wireless client associated to the AP. RX_SIGNAL is the signal power in dBm received by the AP from the transmission of a client STA while RX_MCS is the rate if transmission from the client STA to the AP. Thus, these two readings refer to the uplink: from the STA to the AP. The following readings refer to the downlink path: AP to STA. TX_MCS is the downlink data rate to each STA, TX_RETRY is the percent of packets which were retransmitted, and TX_FAIL is the percent of packets which failed to reach the AP. One of the functions of the `bash` script is to compute these percentages from the raw output in Figure 6. In order to have a sense of the quality of the downlink, we define the transmit packet reception rate (TX_PRR) = 1 - TX_RETRY - TX_FAIL. To preview the use of the key metric PRR-MCS, one can see that the quality of the downlink can be expressed as the summation for all STA of (TX_PRR of the STA) x (MCS of the STA). This total PRR-MCS metric has the advantage of giving more weight to client stations with good link quality, that is, the stations with the poorest downlink connections (low MCS) do not contribute much to the calculation of the total PRR-MCS. This ensures that the quality of the downlink is assessed using the performance statistics of the best associated client stations, as it should be. The total downlink PRR-MCS is the key metric to score one of the genes in a chromosome – the maximum MCS data rate. A modified form of the downlink PRR-MCS will also be used for rating the maximum transmit power and the guard interval. On the other hand, channel frequency and channel setting will be scored using an uplink PRR-MCS with one key difference. Note that the any PRR calculation for the uplink is *only* a prediction since the retransmission and failure rates are not available in the station dump. The only uplink parameters available in the station dump are RX_SIGNAL and RX_MCS. To make such prediction of the uplink packet reception rate, a graph from [15] will be used as shown in Figure 8.

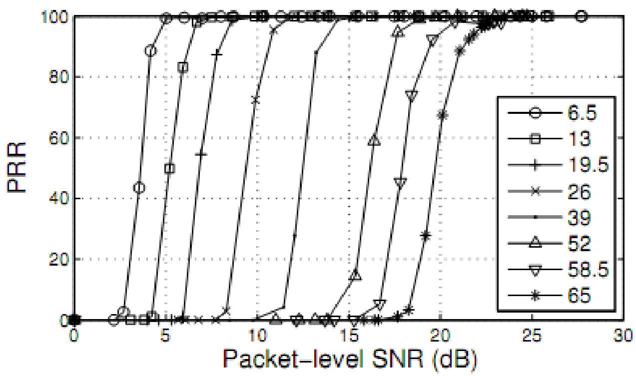

Fig. 8. A reprint of the graph in [15] relating the signal to noise ratio with the packet reception rate. This is a variant of the more well-known bit error rate (BER) versus SNR or Eb/No graphs which characterizes a digital modulation technology. The graph above applies to 802.11n MCS data rates.

One can think of packet reception rate (PRR) as a thread that ties together the indicators of link quality which are the more familiar received signal level (dBm), signal to noise ratio (dB) and interference noise floor (dBm). The graph in Figure 8 is stored as a two-dimensional array in the C++ code. Using this graph, one can either predict the downlink SNR at a client STA given the TX_MCS and the computed TX_PRR. Conversely, one can estimate the RX_PRR given the RX_MCS and the uplink SNR as seen by the AP from each STA. The uplink SNR can be estimated indirectly as the difference between the RX_SIGNAL and the average interference in the channel of operation. As discussed above, the interference information is made available to the fitness function since the output of the channel survey commands are processed into an interference noise floor.

Now, the data gathered and the tools to be used by the GA fitness function are complete and all that needs to be done is to score the genes of a chromosome (Figure 1). The formulas used by the fitness function for scoring each gene are presented in Table 1.

For channel frequency and channel setting, the score computation is straightforward. The gene score is the summation across all the associated client stations of (RX_PRR) x (RX_MCS). RX_MCS is readily available as shown in Figure 7. RX_PRR is determined in the following manner. First the uplink SNR of a client is computed as the client RX_SIGNAL minus the average interference at the particular channel. The interference noise floor is stored in an array for the whole 2400-2483 MHz band (Figure 5). The average interference for a channel frequency and setting combination is the sum of the interference for the channel band divided by the channel bandwidth. For instance, for a channel 1 HT20 setting, the average interference is the sum of the interference array (in mWatts) from 2402 to 2422 MHz divided by 20 MHz. Once SNR is derived from this computation, it becomes a simple matter to locate from the graph the expected RX_PRR given the SNR and the RX_MCS.

The maximum MCS data rate is considered as a proposition for the AP to operate at this rate at the maximum. Note that IEEE 802.11 dynamically changes the MCS data rate based in the link quality as measured by packet losses. The maximum MCS data rate gene is a proposed cap on the current MCS data rate, that is, if the proposed MCS rate is say MCS-6, then the AP cannot transmit at MCS-7 towards any STA at the downlink.

The gene score is computed as the summation for all client STA of the (PREDICTED TX_PRR) x (GENE MCS DATA RATE). The GENE MCS DATA RATE is the rate in Mbps as proposed by the gene in a chromosome, so this value is readily available for use in the computation. he PREDICTED TX_PRR is not specified and is also not available in the station dump command output. From the station dump output (Figure 7), the TX_PRR for each client can be computed using TX_PRR = 1 - TX_RETRY - TX_FAIL. Since, the TX_PRR and the TX_MCS are known, one can use the graph in Figure 8 to estimate the TX_SNR for each client STA. To get the PREDICTED TX_PRR, one again uses the same graph but this time with two new input parameters: an adjusted TX_SNR and the proposed GENE MCS DATA RATE instead of the measured TX_MCS. Take note that the MCS parameter is not the only one that changes. The MCS parameter to be considered to predict the TX_PRR is not the

TABLE 1
FITNESS SCORING FOR EACH GENE OF A CHROMOSOME

Gene	Input Parameters	Score (in unit of % Mbps)
Channel Frequency	**rx_prr** is derived from the graph and the computed snr. **snr** = rx_signal – channel interference	Σ (RX_PRR)(RX_MCS) ALL STA
Channel Setting	**rx_prr** is derived from the graph and the computed **snr**.	Σ (RX_PRR)(RX_MCS) ALL STA
Maximum MCS Data Rate	given the **tx_mcs** and the **tx_prr**., **tx_snr** is derived from the graph using an adjusted tx_snr and the proposed gene mcs, a new predicted tx_prr is obtained from the same graph for each sta	Σ (PREDICTED_TX_PRR) x (GENE_MCS) ALL STA
Maximum Transmit Power	adjusted **tx_snr = tx_snr –** db reduction of transmit power. this adjusted tx_snr is used to obtain the adjusted tx_prr at the sta tx_mcs.	Σ (ADJUSTED TX_PRR) x (TX_MCS) ALL STA
	tx_prr and tx_mcs are readily available from the station dump	For 800 nsec: Σ (TX_PRR)x(TX_MCS) ALL STA For 400 nsec: (1-1.3(1-TX_PRR)) x Σ (TX_MCS) x 1.11 ALL STA

TX_MCS of client, rather it is the MCS setting of the gene. In addition to this, the TX_SNR is also adjusted because changing the MCS data rate entails a change in the effective transmit power. IEEE 802.11n radios are required to comply with decrease in power output when the MCS data rate is increased. The AP used in this experiment has the following characteristic MCS versus transmit power: MCS0 to MCS3 28 dBm; MCS4 27 dBm; MCS5 25 dBm; MCS6 24 dBm; MCS7 23 dBm. The adjusted TX_SNR is equal to TX_SNR + (AP transmit power at the proposed gene MCS – AP transmit power at the TX_MCS). It is expected that this property of the AP of higher transmit power at lower MCS data rates together with the particular link conditions of the associated STA will lead to a trade-off in the choice of the optimum MCS data rate. That is, the highest MCS data rate will not always be the optimum MCS and such result can only be expected when the quality of all or most client STA links are very good. Now, the PREDICTED TX_PRR for each client station is known and one can proceed with the gene score computation.

For scoring the maximum transmit power gene, an adjusted SNR is computed for each STA. This adjusted SNR is simply the TX_SNR used in the MCS gene score above minus the proposed dB reduction in transmit power. From this adjusted SNR and the STA TX_MCS, the PRR vs SNR graph yields an adjusted TX_PRR. The gene score is then computed as the summation across all STA of the (ADJUSTED TX_PRR) x (TX_MCS). Although it is expected that the highest transmit power setting will get the highest score, one can set a margin of allowance for optimization of transmit power. For instance, if the gene score of two transmit power values are within approximately 5%, the lower of the two transmit power values can be given a higher score.

Finally, for the guard interval gene, the short GI of 400 nsec is only recommended if the downlink data rates toward the client STA are high. A rule of thumb from [17] will be used. According to their test measurements, a short GI leads to a 30% increase in error rates while boosting the throughput by 11% (from IEEE 802.11n standard). From this, the gene is scored as follows. For 800 nsec, the gene score is the summation across all clients of (TX_PRR) x

(TX_MCS). The gene score for the 400 nsec case is computed differently as the summation for all clients of (1 – 1.3 x (1- TX_PRR)) x (TX_MCS) x 1.11.

An overview of the multi-objective GA was outlined in Section 4 while a thorough discussion of the fitness function was given in Section 5. The next section compares the self-adaptive AP algorithm with the Automatic Channel Selection (ACS) algorithm used in `hostapd`. The operation and computations involved in ACS are not discussed in much detail and the reader is referred to [16] for a more complete discussion.

VI. EXPERIMENTAL RESULTS

A sample result of `iwlist wlan0 scanning` is summarized in Figure 9. Again a `bash` script is used to create this summary. It averages the values of the signal received from each AP to smooth out sudden spikes in the readings. Also, only the access points with the highest signals are considered – those with signals higher than -90 dBm. Other access points which are too close to the noise floor are ignored since they will not affect the interference noise floor computation in a significant way.

The output of the ACS algorithm and its recommended channel use is presented here. A complete scan from channels 1 to 11 is performed. For each channel, two readings are gathered spaced 5 seconds apart. From the raw data similar to Figure 5, a summary table is obtained (Table 2). From this computation, the unmodified ACS algorithm used by `hostapd` recommends using Channel 1 for HT 20 MHz and Channel 1+5 for HT 40 MHz. If one will check [16], the sample computation led to a choice of channel 13 HT 20 MHz. This is not surprising because the edge channels just summed up 3 values of interference factors versus 5 for the center channels. These results supports the seeming inadequacy of the ACS algorithm, particularly the formulas used to quantify interference. With a modified ACS, using average instead of sum for the interference factors, the recommended channels are channel 2 for HT 20 MHz and 4+8 for HT 40 MHz. Even before a full analysis

TABLE 2
RECOMMENDED CHANNELS OF OPERATION (IN BOLD) BASED ON THE ACS ALGORITHM

Channel prim+seC	setting	band (Mhz)	SUM oF interf factor	AVE oF interf factor
1	HT20	2402-2422	**0.1798**	0.0599
1+5	HT40+	2402-2442	**0.4814**	0.0688
2	HT20	2407-2427	0.2187	**0.0547**
2+6	HT40+	2407-2447	0.5184	0.0648
3	HT20	2412-2432	0.2779	0.0556
3+7	HT40+	2412-2452	0.5745	0.0638
4	HT20	2417-2437	0.3174	0.0635
4+8	HT40+	2417-2457	0.5695	**0.0633**
5	HT20	2422-2442	0.3389	0.0678
5+9	HT40+	2402-2442	0.6527	0.0725
6	HT20	2427-2447	0.3386	0.0677
6+10	HT40+	2407-2447	0.6153	0.0769
7	HT20	2432-2452	0.3558	0.0712
7+11	HT40+	2412-2452	0.5763	0.0823
8	HT20	2437-2457	0.3804	0.0761
9	HT20	2442-2462	0.3888	0.0778
10	HT20	2C447-2467	0.3137	0.0784
11	HT20	2452-2472	0.2767	0.0922

using the self-adaptive AP algorithm, the soundness of the result of the modified ACS specifically 4+8, is supported by the raw data in Figure 9.

```
AP CHANNEL_FREQ SIGNAL SECONDARY_CHANNEL
XX:XX:XX:B6:52:3F 2412 -52.00 no
XX:XX:XX:D4:D9:68 2462 -57.00 below
XX:XX:XX:06:ED:7E 2412 -77.00 above
XX:XX:XX:31:F3:A4 2462 -79.00 below
XX:XX:XX:40:A5:3E 2462 -87.00 below
XX:XX:XX:0A:14:5F 2437 -85.00 no
```

Fig. 9. Sample summary of iwlist wlan0 scanning

Just by observation, the two most powerful interferer access points with signal levels of -52 and -57 dBm are situated at channels 1, and 7+11, respectively. Thus, if one is to choose a 20 MHz channel, channel 4 would be a sound choice indeed. It is now time to apply the algorithm used in the self-adaptive AP.

As discussed in previous sections, the AP will first use the raw data of interferer access points as shown in Figure 9. The resulting interference + noise floor is shown in Figure 10. The AP signal readings in Figure 9 and plotted in Figure 10 are just that – signal strengths. They do not tell what percentage of time that signal strength is present in the air. Thus, the average powers of the interferers are overestimated. Theoretical calculations of packet reception rates (PRR) are found to be too high compared to real measurements when the uplink signal-to-noise ratios are used to get the corresponding PRR from the graph using the interference + noise floor in Figure 10. Thus, the busy time/active time ratio is used as a correction factor to generate a lower interference + noise floor (Figure 11). The signal level received from interferer access points are only those of beacon transmissions which are typically broadcasted by an AP every 100 msec and occupying the medium only for a brief 50 bytes at 1 Mbps data rate. After computing the interference noise floor, the busy time/active time ratio for the chosen channels are expressed in dB and are subtracted from the interference noise floor for those channels resulting to reduction to more reasonable values. After this correction, the uplink signal-to-noise ratios will be higher than before the correction and the theoretical PRR estimates decrease and become closer to real measurements. Note that the uplink signal-to-noise ratio from a client STA is the gathered RX_SIGNAL minus the average interference present in the gene channel frequency-channel setting combination.

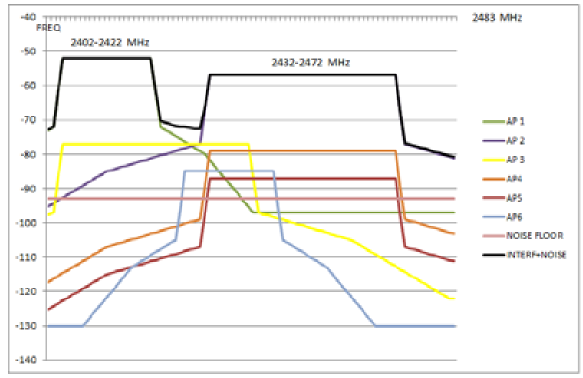

Fig. 10. Interference + Noise Floor (in **black line)** sums up all the interference caused by all the interferer access points in Figure 9 to an initial noise floor of -93 dBm

TABLE 3
CORRECTION FACTORS VERSUS BAND OF FREQUENCIES

Band of frequencies	Chanenl	Correction in dB
2400-2422	1	-10.51
2422-2442	5	-12.28
2442-2462	9	-12.51
2462-2472	11(upper half)	-11.65

The correction factor that will be used to lower the interference + noise floor will use the busy time/active time ratios of channels 1, 5, 9, and 11. The reading at channel 11 is divided by 2 to compensate the fact that the channel 9 scan already covered half of channel 11. The interference factors of these channels are converted into dB using the formula 10log(interference factor). It is actually more proper to call these busy time/active time ratio rather than interference factor since the latter is a terminology used by ACS. The essence of this ratio is that it indicates the percentage of time the channel is occupied or busy. Thus, subtracting this ratio (in dB) from the initial power estimate (in dBm) as shown in Figure 10 gives the corrected interference + noise floor. Table 3 shows the correction factor (in dB) over the band of frequencies and Figure 11 gives the corrected interference + noise floor.

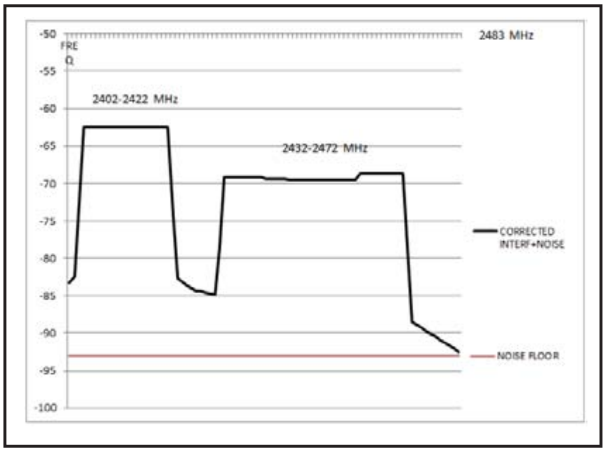

Fig. 11. Corrected Interference + Noise Floor using busy time/ active time ratios of Channels 1,5,9 and 11

The corrected or adjusted interference + noise floor will be the basis for the first series of runs of the GA. At this point, the self-adaptive AP is just exiting STA mode. It has not operated as an AP yet and there are still no connected clients. After initializing the 1^{st} generation of population, the fitness function will evaluate 50 randomly-generated chromosomes with the following objectives: minimize interference, maximize power, and maximize the MCS data rate. The default channel setting of HT 20 MHz and the guard interval of 800 nsec will also be given higher scores. This way, the device can immediately start operating as an AP and serve wireless clients. In the context of the overall program flow, the function objective_channel() in the C++ code has finished its job, that is, to construct the interference + noise floor (Figure 11) including the correction provided by the busy time/active time ratio. At this stage of the code, and for simplicity, PRR-MCS has not yet been introduced into the fitness calculations since the AP has not yet gathered client STA statistics. The channel frequency and channel setting configurations are initially scored using the average interference in their band of operation. The average interference is equal to the area under the curve in Figure 11 over the band of operation (e.g. 2402-2422 for channel 1 at HT 20 MHz) divided by the channel setting bandwidth (e.g. 20 or 40 MHz). The use of a simpler fitness scoring for the initial GA run does not devalue the concept of PRR-MCS. One can assume an RX_SIGNAL and an RX_MCS from a hypothetical wireless client STA and get a corresponding SNR by subtracting the average interference from the RX_SIGNAL. Then the PRR vs SNR graph can be used to predict a RX_PRR and multiplying this by the RX_MCS yields a PRR-MCS metric which can be used for fitness scoring. However, it is obvious that such fitness scoring will lead to the same solution because the variables are held constant except for the average interference, which varies with the channel band. Table 4 summarizes the result of the initial set of GA runs using the average interference fitness scoring scheme. Note that the fitness score is not yet in terms of PRR-MCS.

The result in Table 3 clearly showed that the solution converged to a channel frequency of 5 and channel setting of HT 20 MHz. This band covers 2422 to 2442 MHz. Figure 12 compares the recommendations of the self-adaptive GA versus the best recommendation of ACS (4+8 or 2417-2457MHz) and clearly demonstrates that the former outperforms the latter in interference mitigation via channel selection. ACS clearly missed channel 5 which sits in a region of lowest interference. There are also other recommendations from the self-adaptive GA, with fitness scores of 138, to use channels 7 to 11 using 20 or 40 MHz but these chromosomes were removed by the 5^{th} generation. Lastly, channels 12 and 13 were not included in the calculations since the AP hardware used does not support those bands.

TABLE 3
MOST FIT CHANNEL CONFIGURATIONS USING INITIAL SET OF GA RUNS (6 GENERATIONS)

LAST SEEN IN generation	channel freq	channel setting	FITNESS SCORE/ REMARKS
5^{th}	6	HT20	140
5^{th}	9	HT40-	140
6^{th}	5	HT20	144(CONVERGED SINGLE SOLUTION)

At this point, the AP is serving wireless client stations and it will continuously gather wireless client statistics. Again a `bash` script is used outside of the C++ code. The code can access the `bash` script using a system() call and it will parse a text file generated by the `bash` script. The script will also handle averaging the readings, specifically signal and MCS, in order to smooth out any variations. For the succeeding discussion, the associated client STA statistics are those shown in Figure 7.

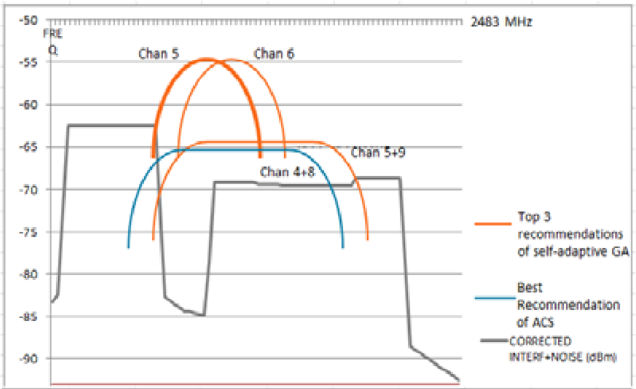

Fig. 12. Top 3 recommendations of self-adaptive GA versus the best recommendation of ACS algorithm

In the context of the program flow, the function objective_station_dump() will take over in a way similar to what objective_channel() did with the scan of interferer access points. The objective_station_dump() function will process the gathered wireless client statistics and most importantly, derive the PRR-MCS values for the 5 chromosome genes. Only after this function is done with these computations will evaluate_fitness() be called again using the revised objectives of minimize interference, optimize transmit power, and optimize data rate. The details of objective_station_dump() was already discussed extensively in the previous section. For this paper, the fitness score of a chromosome is computed as just the sum of the fitness scores of each gene and all are in units of PRR-MCS (% x Mbps). Table 4 shows the top 10 solutions of a sample run by the 6th generation.

Table 4 serves to illustrate the variation in the most likely candidates from the whole search space. By the 10th generation of the sample run, most of the 50 chromosomes in the population have either 5 or 5+9 as channel frequencies, and only 8% of the chromosomes have channel 6. Transmit power reductions are either 0 or 1 dB with 72% of the chromosomes recommending a reduction of 1 dB from the maximum transmit power. Thus optimization of transmit power can be achieved by the multi-objective GA and is often recommended as a better option than simply setting the transmit power at maximum. The more aggressive guard interval of 400 nsec is only selected only 18% of the time. Finally, the MCS data rate is more diverse. This is a good indication that the multi-objective GA (MOGA) is adapting to the wireless conditions of the client stations. The advantage of a lower data rate in downlink is that stations can connect at lower error rates. From the preceding discussion, it is evident that the MOGA for the self-adaptive Access Point is well suited for such interplay of wireless parameters and network

TABLE IV

Most Fit Chromosomes by the 6th Generation

chan freq	Chann setting	band (Mhz)	max data rate mbps	less tx pow (db)	GI	FIT score
5+9	HT40+	2422-2462	52x2	0	800	430
5+9	HT40+	2422-2462	52x2	0	800	430
6	HT20	2427-2447	39	**0**	400	424
5+9	HT40+	2422-2462	6.5x2	0	400	423
6	HT20	2427-2447	39	1	800	422
5+9	HT40+	2422-2462	52x2	1	400	416
5+9	HT40+	2422-2462	39x2	1	400	416
5+9	HT40+	2422-2462	52x2	**1**	400	416
5+9	HT40+	2422-2462	39x2	1	400	416
5	HT20	2422-2442	52	2	400	410

conditions. Figure 13 closes this section with a graph of the distribution of maximum MCS data rate for the 50-chromosome population by the 10th generation.

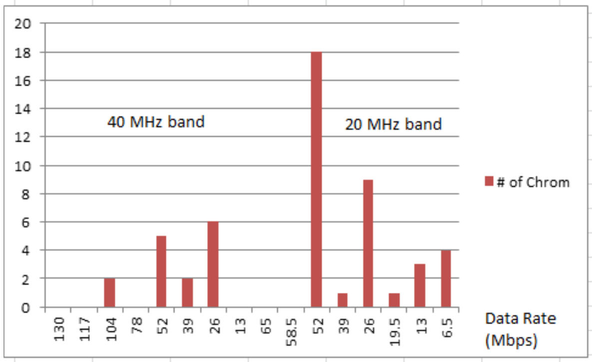

Fig. 13. Distribution graph of instances of certain MCS data rates across a population of 50 chromosomes by the 10th generation

VII. Conclusion and Future work

Variations on the fitness score or the GA operations are possible extensions to this work. A weighted sum approach for the fitness function or a tournament approach in the GA selection are just some of the enhancements worth exploring in the future. Such enhancements are well supported by the most important contributions of the current paper, namely: (1) the single-metric of PRR-MCS for scoring the different genes, (2) the normalization across multiple objectives or dimensions of interference mitigation and optimization of bandwidth, data rate, power, and guard interval, and (3) the implementation of multi-objective genetic algorithm (MOGA) to make a WLAN access point self-adaptive to dynamic wireless network conditions and to multi-radio ecosystems (e.g. HetNets).

Another important next step for this research is the actual compiling of the C++ code into a Linux-based WiFi Access Point. Important requirements for such endeavor have already been identified and discussed in this work. In order to fully implement a stand-alone daemon based on the C++ code, thresholds should be identified such as what level interference, error rates, or PRR-MCS will trigger a restart of the GA or the program. Finally, possible use of Heuristic Search Technique or Multi-Mode Self-Adaptive (MMSA) algorithm can be explored.

Acknowledgment

The authors would like to thank the Linux and open-source community whose programs were used as cited in different parts of this paper. Thank you for powering not only the appliance used for the experiments but for powering the Internet as well.

References

J. G. Andrews, "Seven Ways that HetNets Are a Cellular Paradigm Shift" IEEE Communications Magazine, vol. 51 no. 3, March 2013. pp. 136—144.

A. Zubow, and R. Sombrutski. "Reinvestigating Channel Orthogonality - Adjacent Channel Interference in IEEE 802.11n Networks," Tech. Republic, 2011.

T. W. Rondeau, B. Le, CJ. Rieser, and C. W. Bostian, «Cognitive Radios with Genetic Algorithms: Intelligent Control of Software Defined Radios," Proceeding of the SDR 04 Technical Conference and Product Exposition, SDR Forum 2004.

A. Garcia-Saavedra, B. Rengarajan, P. Serrano, and X. Costa-Pérez, «SOLOR: Self-Optimizing WLANs with Legacy-Compatible Opportunistic Relays,» IEEE/ACM transactions on Networking, vol. 20 no. 10.

H. B. A. Sidi, Z. Altman, and A. Tall, «Self-Optimizing mechanisms for EMF reduction in heterogeneous networks,» 2014 12th International Symposium and Workshops on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), Hammamet, Tunisia, May 2014.

R. Pries, D. Staehle, B. Staehle, and P. Tran-Gia, «On Optimization of Wireless Mesh Networks using Genetic Algorithms,» International Journal on Advances in Internet Technology, vol. 3 nos. 1 & 2, 2010.

Y. S. Chia, Z. W. Siew, H T. Yew, S. S. Yang, and K. T. Z Teo, «An Evolutionary Algorithm for Channel Assignment in Wireless Mobile Networks,» ICTACT Journal on Communication technology, vol. 3 no. 4, December 2012.

X. Fu, A. G. Bourgeois, P. Fan, and Y. Pan, «Using a genetic algorithm approach to solve the dynamic channel-assignment problem,» International Journal of Mobile Communications, Vol. 4, No. 3, 2006, pp. 333-352.

Y. Ding, Y. Huang, G. Zeng, and L. Xiao, «Channel Assignment with Partially Overlapping Channels in Wireless Mesh Networks,» Proceeding WICON '08 Proceedings of the 4th Annual International Conference on Wireless Internet, Article No. 38, 2008.

B. Jalili, and M. Dianati, «Application of Taboo Search and Genetic Algorithms in Planning and Optimization of UMTS Radio Network,» Proceeding IWCMC '10 Proceedings of the 6th International Wireless Communications and Mobile Computing Conference, pp. 143-147.

D. Cacciani, F. Garzia, A. Neri, and R. Cusani, «Optimal Territorial Resources Placement for Multipurpose Wireless Services Using Genetic Algorithms,» Wireless Engineering and Technology, vol. 2 no. 3, July 2011, pp. 184-195. Available: http://www.SciRP.org/journal/wet

B. Paikaray, “Vertical Handoff Decision Based On Genetic Algorithm in 4G Network,” CoRR abs/1209.4968, Cornell University Library, Ithaca, NY, 2012. [Online]. Available: http://arxiv.org/pdf/1209.4968v1.

“The Effects of Adjacent Channel Rejection and Adjacent Channel Interference on 802.11 WLAN Performance,” Texas Instruments Whitepaper SPLY005, November 2003.

J. Zhang, K. Tan, J. Zhao, H. Wu, and Y. Zhang, “A Practical SNR-Guided Rate Adaptation,” INFOCOM 2008. The 27th Conference on Computer Communications. IEEE, pp. 13-18 April 2008.

D. Halperin, W. Hu, A. Sheth, and D. Wetherall, “Predictable 802.11 packet delivery from wireless channel measurements,” ACM SIGCOMM, 2010.

“ACS - Linux Wireless,” The Official Linux Wireless Wiki. Web. <http://wireless.kernel.org/en/users/Documentation/acs>.

“Wireless Reliability - Guard Intervals,” The InterOp Insider. University of New Hampshire InterOperability Laboratory. Web. <https://www.iol.unh.edu/blog/2014/6/16/>.

R. Ligeiro and H. Miranda, “An analysis to the deployment of access points using genetic algorithms.” [Online]. Available: http://www.di.fc.ul.pt/~hmiranda/papers/ligeiro06-slides.pdf

K. Ohnishi, K. Tsukamoto, S. Kashihara, & Y. Oie. “Self-configuration of wireless access points based on mechanisms of biological development and evolution,” Self-Adaptive and Self-Organizing Systems Workshop (SASOW), 2010 Fourth IEEE International Conference, pp. 17-25.

T. Vanhatupa. Wi-Fi Capacity Analysis for 802.11 ac and 802.11 n: Theory & Practice. Ekahau Inc, 2013.

A. Riedl, “A Versatile Genetic Algorithm for Network Planning,” Proceedings of EUNICE, vol. 98, September 1998, pp. 97-103.

J. Kim and J. Lin, “Comparison of self-adaptive wireless networks using mobile base stations and mobile access points,” Wireless and Microwave Technology, 2005, pp. 108-110.

D. Turner, S. Savage, and A. C. Snoeren, “On the empirical performance of self-calibrating wifi location systems,” 2011 IEEE 36th Conference on Local Computer Networks (LCN), October 2011, pp. 76-84.

A. Zubow, and R. Sombrutzki, “Adjacent channel interference in IEEE 802.11n,” IEEE Wireless Communications and Networking Conference (WCNC), April 2012, pp. 1163-1168.

P. Miklavcic, “On the number of non-overlapping channels in the IEEE 802.11 WLANs operating in the 2.4 GHz band,” Elektrotehniski Vestnik, 2014, 81(3): 148.

S. Vigneshwaran, and C. Selvaraj, “Implementation of WLAN N and Estimation of Co Channel and Adjacent Channel Interference,” J Comput Eng Inf Technol 5: 2, http://dx.doi.org/10.4172/2324, 9307, 2.

G. K. Audhya, K. Sinha, S. C. Ghosh, and B. P. Sinha, ”A survey on the channel assignment problem in wireless networks,” Wireless Communications and Mobile Computing, 11(5), 2010, pp 583-609.

J. Cheeneebash, J. Antonio Lozano, and H. Coomar Shumsher Rughooputh. “A survey on the algorithms used to solve the channel assignment problem,” Recent Patents on Telecommunication, 1(1), 2012, pp. 54-71.

I. Amro and W. Ghanem, “GSM Automatic Frequency Planning with the Aid of Genetic Algorithms,” Fifth International Conference on Genetic and Evolutionary Computing (ICGEC), August 2011, pp. 73-76.

S. L. V. S. Jyothi, and S. S. Gowri, “Modified Genetic Algorithm for Channel Allocation Problem.”