Self-Optimization of Software Defined Radios Through Evolutionary Algorithms

Zubair Shaik, André Puschmann and Andreas Mitschele-Thiel
Integrated Communication Systems Group, Technische Universität Ilmenau, Ilmenau, Germany
Email: [zubair.shaik, andre.puschmann, mitsch]@tu-ilmenau.de

***Abstract*—This paper presents a framework for building software-defined radios that are able to self-optimize their parameters using evolutionary algorithms. The framework has been implemented using the DEAP library for Python, which is based on the Genetic Algorithms (GAs). The paper discusses the overall system architecture and presents a system prototype that has been employed to optimize radio transmission parameters in an unknown radio environment in order to maximize the achievable throughput. Although GAs have been used before for optimizing the radio parameters of Software Defined Radios (SDRs), they have been limited to the number of parameters given as an input to the GA. The proposed algorithm is much more generic and comprehensive to utilize the advantages of genetic algorithms, by providing the flexibility to include any of the parameters of the configuration of the SDR, which needs to be optimized through the GA. Moreover, the entire project is based on open-source solutions. The current prototype targets Iris-based SDRs. However, as the entire software employs standard components for interfacing the SDR, it can easily be ported to GNU Radio or other SDR frameworks. We will also present preliminary results that have been obtained through over-the-air experiments in which we optimized different power parameters, modulation, coding schemes, etc., in an unknown radio environment.**

Keywords*—*Software Defined Radios (SDRs), Distributed Evolutionary Algorithms in Python (DEAP), Iris, Genetic Algorithms (GAs), Multi-Objective Genetic Algorithms (MOGA)

I. Introduction

Wireless communication systems are built in a layered fashion comprising a multitude of protocols and services. The wireless signals are too volatile to its surroundings and to figure out the optimal working conditions is a challenging task. Software defined radios (SDR) provide the flexibility to implement these different services on a software platform, be it the physical layer modules of modulation and coding or the MAC layer protocols. The configuration space of an SDR is too huge, and it becomes difficult for a regular user to figure out a good working set of these settings. We utilize the flexibility of an SDR to create a self-optimizing radio that intelligently adapts to its surroundings and works in an optimal scenario.

Generally, too many parameters guide the working of a wireless radio. The solution space includes many possible combinations of the parameters that govern the wireless radio, like frequency, bandwidth, transmit/receive power, modulation and coding schemes, gain values etc. These parameters could exist from the application layer to the MAC and physical layer. The number of combinations of these parameter values would be in millions and an exhaustive search would take years to figure out the best possible combination. The objective of the cognitive radio is to find tractable solutions for optimizing the radio link but in a desired time applicable to the user. Various algorithms have been proposed over the years to optimize the parameter configuration. Doerr *et al.* [1] classified these algorithms into four main categories: genetic algorithms, game theory, rule-based reasoning and neural networks. We choose genetic algorithm as it is good for large solution space and gradually evolves its solution set based on the experiences during its evolution.

This paper focuses on the self-optimizing radio which employs genetic algorithm to adjust its radio parameters. The radio engine developed is based on the following open-source solutions: (1) *DEAP* [2], an evolutionary computation framework for Python, (2) *Iris* [3], an application framework for building reconfigurable SDRs, and (3) *OSPECORR* [4], a middleware used to connect the individual software components together. An early prototype of the platform has been presented as a talk at FOSDEM 2016 [5].

The remainder of the paper is sectioned as follows: In Section II, we briefly describe genetic algorithms and give an overview about multi-objective decision making. Section III describes the application of GAs to software defined radios. Section IV-A presents the design and the component structure of the proposed system, while describing the employed genetic algorithm using DEAP. Section V contains the experimental results of the algorithm. Finally, we summarize our findings and conclude the paper in Section VI.

II. Genetic Algorithms

Genetic algorithms (GAs) are heuristic search algorithms inspired by biological observations. They adapt based on the evolutionary ideas of natural selection and genetics. As such, they belong to the class of evolutionary algorithms. The techniques used in GAs are based on the natural evolution, such as selection, mutation, crossover, and inheritance. These heuristics are frequently being used for optimization problems, as they exploit the information during their evolution to direct the search into a solution space that generally gets better each time. In particular, where the state space is large, multi-modal or a n-dimensional surface, GAs provide significant optimization solutions when compared to typical search techniques such as depth-first, breadth-first or praxis [6].

GAs are based on two components. The first component is the *genetic representation* of the possible solution space, the so-called chromosomes. This physical representation of a particular solution of the algorithm is modeled through bit strings of variable length that contain the value of each parameter, i.e., the gene. The second component is the fitness

function that evaluates each solution set. It can be a simple formula or a statistic from a complex simulation.

The algorithm starts by randomly generating an initial population, which is a pool of individual potential solutions. It evolves through three basic operators of selection, crossover and mutation. Selection involves giving preference to individuals, which allows their genes to be passed on to their future offspring. Each individual is associated with a fitness value that can be determined by the fitness function. This ensures that the individuals with a good fitness value quickly dominate the population. Mutation and crossover are variation operators used to create different children, but still related to their parents. Mutation is done to one individual at a time, and involves bit flipping of their bit strings, where as crossover operator takes two or more parents to make a child which is a combination of them. The probability values of crossover and mutation are given as inputs to decide the intensity of variation of the offspring. Survivor selection mechanisms based on the fitness and age of the parents and offspring determine which individuals are carried on for the next generation.

The GA, using the operators of selection, crossover, and mutation gradually evolves through generations to converge towards a global optimum solution [7]. A termination condition has to be applied to the genetic algorithm, to prevent it from an infinite or an exhaustive search. The termination condition can be based on the number of generations or an fitness threshold. This could be either an absolute value or when there is minimal change in the fitness values of the population. Figure 1 illustrate the basic process flow of a GA.

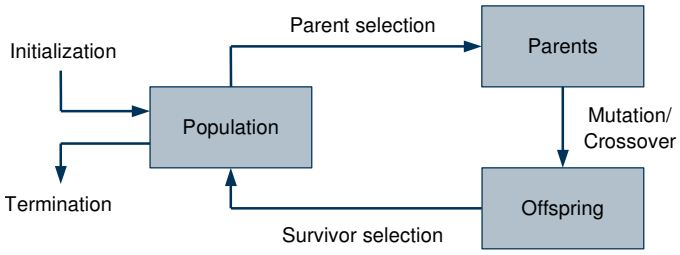

Figure 1. GA process flow (adopted from [8]).

GAs have been mainly applied to problems with a single objective. However, most of the real-world problems are multi objectives. For example in the case of wireless communication systems, it's not always the maximum throughput alone that's an objective, but along with it, the transmit power, frame error rate, etc., also need to be considered. In Multi-objective Genetic Algorithms (MOGA) [7], weights can be attached to the individual objective functions, directing the results to achieve the maximum/minimum values as required.

To give an example, consider two objective functions of a GA, $f_1(\cdot)$ and $f_2(\cdot)$. Further assume that $f_1(\cdot)$ is to be maximized and $f_2(\cdot)$ to be minimized. In order to optimize for both objectives, MOGA uses a weighted sum of multi-objective functions to form one scalar fitness function. With the search evolving in different directions, a set of Pareto-optimal solutions are achieved, with individuals of high fitness values selected from each set to form the elite of the next generation. Pareto optimality is a state of allocation of resources where any change beneficial to one individual is detrimental to one or more others. When no further pareto improvements can be made, an allocation is called Pareto-optimal [9]. Figure 2 illustrates the direction of search of a MOGA with four non-dominating solutions, all laying on the Pareto front.

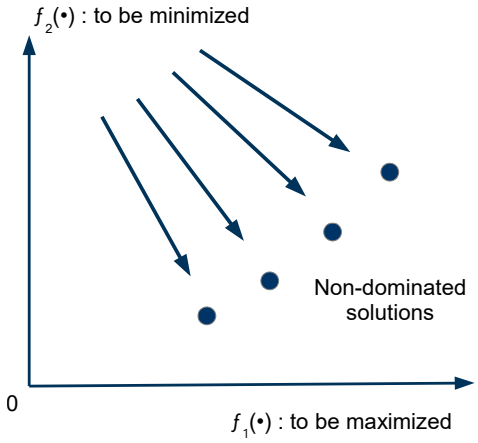

Figure 2. Directions of the search in MOGA (reproduced from [7])

III. Application of Genetic Algorithms to SDRs

For a successful wireless transmission, the radio must be configured according to the channel conditions. The channel conditions are very diverse in nature and the impact of them on the radio transmission is very unpredictable. Numerous solutions have been proposed over time to stabilize the radio transmission, which includes better modulation and coding techniques, the use of right power and gain values, frequency selection, symbol rates etc. Looking at the huge solution space, it turns out to be a very daunting task to figure out which configuration works best. The idea is to let the GA decide and figure out the optimum solution.

By providing the range of feasible values for all the parameters that effect the radio transmission, even in an unknown radio environment, the algorithm takes a comprehensive control and starts the optimization process through gradual evolution. The time required to arrive at a good optimum solution is directly proportional to the solution space.

The parameters of both the transmitter and the receiver that are to be optimized form the genotype of an individual, as shown in Figure 3. Note that the genes of the chromosome may have different lengths, depending on how many feasible values exist per parameter.

The initial set of individuals are randomly generated by assigning random values to each of the parameter from their corresponding solution sets. Traffic conditions of the radio transmission govern the fitness evaluation function of the GA. The termination condition is controlled by the evolved number of generations. The parameters of the GA, like number of individuals, number of generations, the crossover and mutation probability are configurable, so as to let the user control the GA as required.

Probably among the first to apply GA to SDRs were Rieser and Rondeau. Rieser *et al.* [10] provide a cross-layer mechanism to deliver the requested Quality of Service (QoS) through Cognitive System Monitor (CSM), by controlling

parameters like power, frequency, modulation type, FEC and TDMA timeslot ratios. Building upon that work, Rondeau *et al.* [11] point in the direction of controlling more parameters by providing dynamic fitness selection and evaluation. Their proposed Wireless System Genetic Algorithm (WSGA) is a MOGA based algorithm to realize cross-layer optimization of a radio. The parameters of PHY and MAC layer form the genes of a chromosome, and their analysis is done through fitness functions defined by evaluation of the radio channel. These fitness functions are dynamically linked from the database, so as to add and weigh them in the fitness function for the evaluation of the wireless link.

The algorithm proposed in this paper is inspired by Rondeau *et al.* [11]. It extends the dynamic functionality by including not just the fitness function but the configuration space as well. Any configurable parameter of the SDR, be it from the application, MAC, or PHY layer, can be dynamically included to form the chromosome of the individual. Furthermore, because it's not just the radio properties of the transmitter that affect the wireless link, the receiver functionality too has to be optimized. The current algorithm includes the configuration of the transmitter as well as the receiver in the chromosome and optimizes both of them simultaneously to achieve the best results.

Very recently, Kozel [8] has employed GAs to optimize digital modulation schemes, i.e., to find a constellation that works best for the given radio conditions. However, this work is only based on simulation and does not consider any over-the-air experiments.

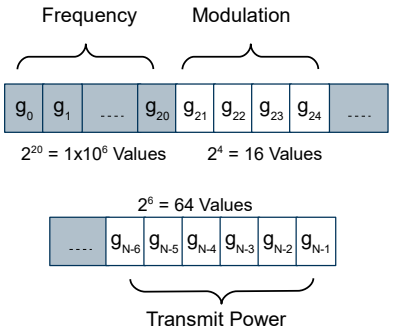

Figure 3. Representation of a chromosome containing genes with a variable length (reproduced from [12]).

IV. Self-Optimization Framework

This section presents the proposed framework for enhancing existing SDRs in order to allow them to self-optimize their communication parameters in real-time. We will first explain the high-level system design and then provide details about our prototype implementation based on Iris.

A. System Design

The system architecture consists of two main components: the optimizer client and the optimizer controller, as shown in Figure 4. The client and the controller represent the transmitter and the receiver of the radio link respectively. The optimizer controller is the basic component that governs the complete process of optimization. The client is a passive module following the protocol set by the controller, executing its commands, and sending back the requested statistics. The controller has been developed in a way that it can be run independently from the transmitter and the receiver, e.g., on a different host machine.

The optimizer controller and client are connected over separated control and data interfaces. This has two main advantages. First, signaling messages sent over the control interface do not negatively effect measurements on the data interface. Second, and more importantly, a separated control interface allows to also configure radio parameters that result in a non-working communication link - something that may happen anytime during the optimization - without having to worry about how to detect and repair such a situation during the optimization procedure.

To evaluate the fitness level of each configuration of the individual, the results of the data transmission are to be computed. *nuttcp* tool which is used to generate the data traffic provides the results of the data transmission in terms of throughput. *OSPECORR* provides the means to collect the physical or MAC layer properties like EVM, RSSI, etc., which are accumulated through the time of data transmission. These computed results are sent to the optimizer controller through the control interface.

Figure 4 illustrates the core building blocks of the system as well as the specific components used for the prototype implementation, which is described below.

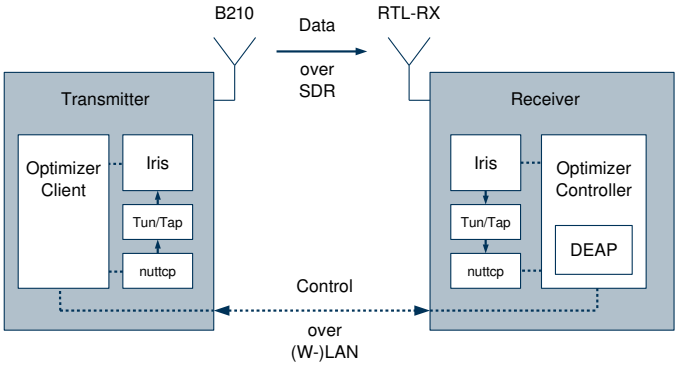

Figure 4. System Design

B. Principle of Operation

The message sequence chart of the controller protocol is shown in Figure 5. The optimization starts by opening a TCP connection between the optimizer controller and the optimizer client, over the control interface. The optimizer controller configures the client with the required settings for the data transmission of *nuttcp* and initiates the GA to continue its evolutionary process. During the GA, the controller is tasked with configuring the settings of each individual on the client, to signal the data transfer and then to collect the results of the data transmission. When the termination strategy decides the end of optimization, the controller configures the best selected configuration and terminates the optimization.

The GA itself is realized by leveraging *DEAP*, an "evolutionary computation framework for rapid prototyping and testing of ideas" [2]. The algorithm initially registers the modules required for the GA, like individual, population, evaluate,

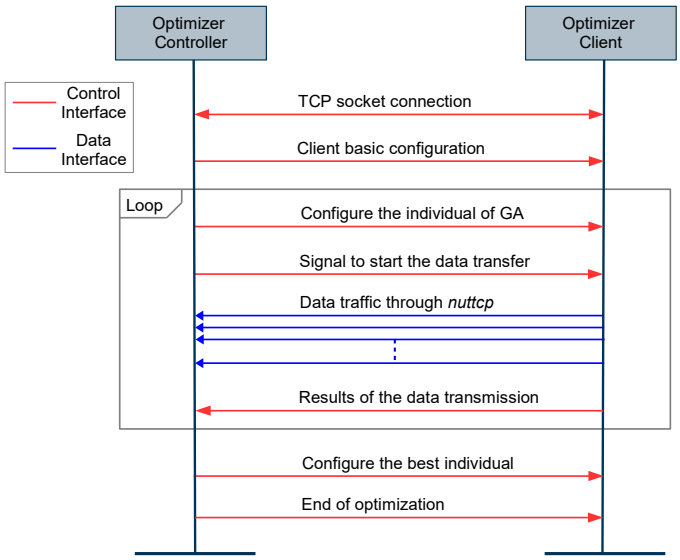

Figure 5. MSC of the optimization protocol

Table I. DEAP OPERATORS

GA Parameter	DEAP Operator
Individual	initCycle
Population	initRepeat
Mutation	mutFlipBit
Crossover	cxTwoPoint
Select	selNSGA2

mate, mutate, and select. Table I shows the operators used through the evolutionary tools of *DEAP*. All the modifiable parameters which are to be optimized form the genotype of an individual, as mentioned in Figure 3. *DEAP* provides a sorting based multi-objective evolutionary algorithm called as Non-dominated Sorting Genetic Algorithm II (NSGA2). NSGA2 alleviates the main problems faced by MOGA, which are computational complexity, nonelitism approach, and the need for specifying the sharing parameter, as explained by Deb *et al.* [13]. Their results prove that NSGA2 is able search better solutions towards the optimum pareto front.

The algorithm starts its run by creating a set of individuals by randomly selecting the parameter values in their genotypes and calculating the fitness of each individual. After the initialization, the evolution procedure begins, going through the mechanism of crossover and mutation of the offsprings and computing their fitness. The evolution continues till the terminate function decides to stop the evolution. In every evolution, a set of offsprings are formed by copying the individual chromosomes from the current generation. The selection procedure from the parents is based on NSGA2, as described before. In this algorithm, an equal number of offsprings are generated as that of parents. These offsprings now go through variation operators of mutation and crossover, to create individuals with different properties, but which are still related to their parents. In crossover, individuals are created by a combination of chromosome bit-strings of two parent individuals, by calling the mate operator. The offsprings then go through the mutation operator, which flips the bit-strings of their chromosomes to make new individuals. Crossover and mutation probability parameters decide the intensity of variation.

Fitness values of these newly created offsprings are calculated and they go through the selection phase of NSGA2 again. The population for the next generation is created by selecting the individuals among the parents and the offsprings, which have the best fitness values. The next step is the termination test, which decides if the evolutionary process has to be stopped or continued. In our algorithm, we stop the process after a fixed number of generations as selected by the user. But the logic can be extended to include conditions like, minimal change in the fitness values of the best individual in each generation, or until an absolute fitness threshold is reached. If the termination test is not met, the process continues to create new generations.

C. Prototype Implementation

The prototype system consists of a uni-directional communication link between a SDR transmitter and receiver. We employ Iris as the underlying SDR framework and utilize the reconfigurability to allows the GA to reconfigure the radio parameters during run time. The optimizer controller is an extension of the Python-based *pySysMoCo* module of *OSPECORR* [4]. *pySysMoCo* is a graphical application to monitor and control various parameters of the SDR. In particular, this allows to select the parameters and settings for the optimization algorithm through a graphical user interface. Furthermore, it allows to display the fitness of the optimization while it is running.

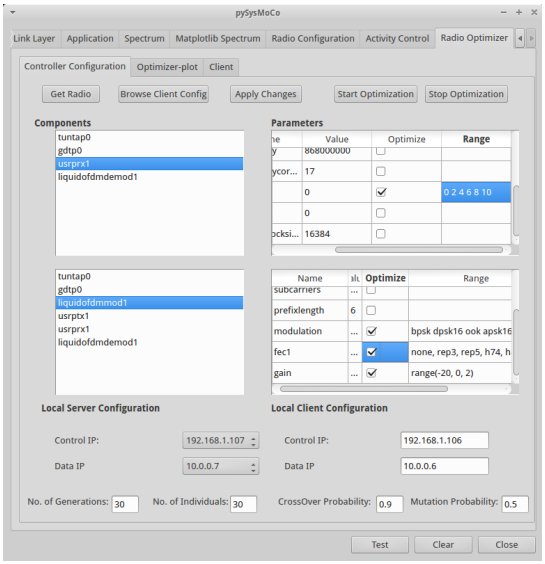

Figure 6. Configuration of the algorithm through the controller GUI.

Figure 6 shows a screenshot of the controller settings of *pySysMoCo*. It allows to select the parameters and their range, i.e., the possible values that each parameter can have, for the transmitter and receiver independently. Note that certain parameters, like operating frequency, number of subcarriers, etc., need to have identical values for both the transmitter and the receiver to work properly. The implemented protocol, however, intelligently handles such conditions as well. Furthermore, the GUI also allows to set the IP addresses of the controller and

client interface as well the core parameters of the GA, e.g. mutation rate.

The fitness of the system evaluated by generating constant bit rate UDP traffic using *nuttcp*. Throughput and data loss form the two objective functions of fitness evaluation, with the former to be maximized and the later to be minimized. In addition to that, we also employ error vector magnitude (EVM), a physical layer metric to quantify the performance of a digital communication system, as an additional objective function.

For the fitness value computation, the optimizer controller sends a frame to the optimizer client over the control interface, to execute *nuttcp* with the specified configuration. After the *nuttcp* has finished, the optimizer client sends the results to the controller. The controller then computes the fitness value of each individual in the population.

In our prototype, the control interface is a realized over a TCP connection over WLAN. The data interface represents the wireless connection that needs to be optimized, i.e., the SDR link.

V. Experimental Evaluation

In order to evaluate the effectiveness of the GA, we have carried out a large number of experiments in our lab. For all experiments discussed in this paper, we employ a USRP B210 on the transmitter side and a RTL-SDR dongle on the receiver. The RTL-SDR is based on the Realtek RTL2832U chip, an inexpensive USB dongle which has a highest theoretical sample rate of 3.2 MS/s.

Table II. Adaptable parameters of SDR

Tx/Rx	Parameter	Range
Tx	Modulation	bpsk, ook, dpsk, apsk16, apsk 32, qpsk, ...
Tx	Coding	none, rep3, rep5, h74, ...
Tx	software gain	-20 to 0 in steps of 2
Tx	usrptx gain	50 to 90
Rx	rtlrx gain	0 to 10

Table II shows the adaptable parameters selected on the transmitter and receiver configuration to run the experiment. The first column represents if the configuration is of the transmitter or the receiver, whose parameter configuration is handled accordingly by the optimizer controller.

In general, the GA is able to include any of the exposed radio parameters in the optimization procedure. In our experiments, however, we limited the number of parameters to those mentioned in Table II. Some of those parameters, if configured on transmitter, the same value has to be configured on the receiver too. The optimizer algorithm checks for such parameters and configures them accordingly.

The list of modulations and coding is much more than what is mentioned in the table. As has been mentioned above, in order to evaluate the fitness we consider the data throughput acquired by running *nuttcp* as well as the EVM obtained from the physical layer. Of those two functions, throughput represents the function to be maximized and the absolute value of EVM is to be minimized. For the multi objective selection, throughput is given the higher weight than that of EVM. To calculate the baseline to be used later for comparison, we manually configure the radio to use 16-QAM as modulation scheme and known values of power and gain. With this setup, we were able to achieve a maximum throughput of 1.7 Mbps.

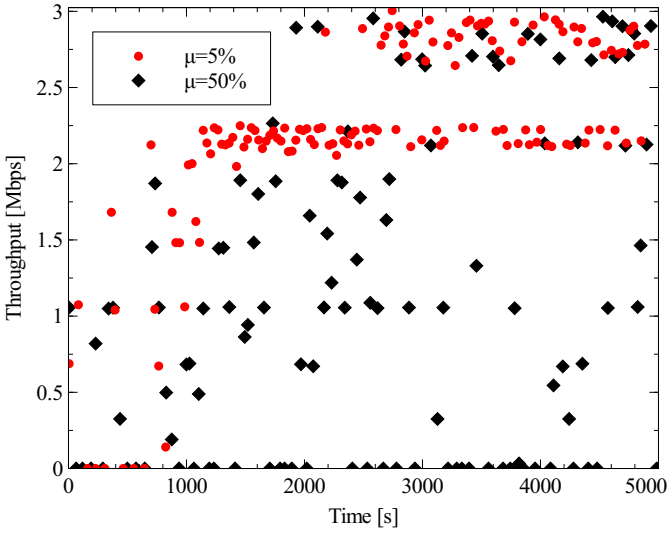

Figure 7. Fitness of each individual generation over time as a function of the mutation rate (μ).

Table III. GA Parameters

Parameter	Values:Set 1	Values:Set 2	Values:Set 3
Mutation Rate	5%	25%	50%
Crossover Rate	90%	90%	90%
Population Size	30	30	30
Max Generations	30	30	30

The parameters of GA used in the experiments are mentioned in Table III. They are those used in [11]. But we run the experiments by increasing the mutation probability in each run. This allows for the increase in the variation of the offsprings from their parent chromosome, to discover better solutions. Initially, we run the experiment based on the parameter values from the column of set 1 and 3. In this experiment, the algorithm focuses on maximizing the fitness function of just the throughput.

Figure 7 shows the throughput values of the individuals produced in each generation over time. It can be observed that with a lesser probability of mutation, the throughput values of the individuals are less scattered, and stay close to that of their parents. In the initial stages, the throughput of the individual with the best fitness value, increases by a good amount, but starts saturating with the increase in time, leading us through the termination condition.

Although the test stops after 30 generations, we can observe that the high throughput has been attained in the halfway of the process. But with an increased mutation probability of 50% from the set 3, the algorithm tries with more variation in the radio parameter values while getting different throughput results than that of the parents. But each generation when formed consists of the elite individuals from the set of parents and the offsprings. Such a case, with a higher rate of mutation, allows variation in the parameter values, figuring out the results of some radio parameter combinations that could have been missed. Although in this experiment, both runs produce

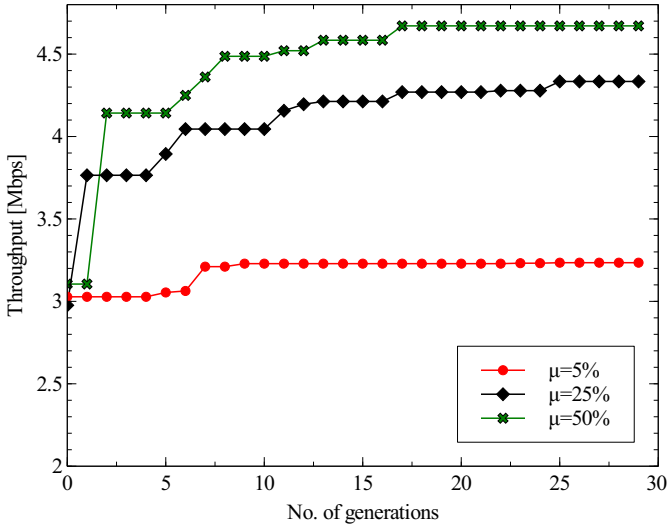

Figure 8. Fitness of the best individual after each generation over time as a function of the mutation rate (μ).

the same maximum throughput of 3 Mbps. The experiment takes beyond 5000 s to run for 30 generations, and that is because of *nuttcp*'s timeout whenever the algorithm does not produce an useful configuration, i.e., produces 100% packet loss. The transmitter waits for a relatively long amount of time before trying out the next solution. This does not affect the optimization as such but can be changed by either modifying the timeout value or by using another application for traffic generation.

In the next experiment, EVM was included along with throughput in the fitness functions. This experiment was run based on all three parameter sets of the GA from Table III. Figure 8 shows the throughput values of the best individual in each generation. To compute the theoretical maximum throughput for the setup used in the experiment seems difficult. Comparison with other systems/approaches is difficult too because most radios use very different system parameters, i.e. bandwidth, frequency, available modulation and coding schemes, etc. Using the lower sample rate of RTL-SDR of 3.2 MS/s among the two radios and with a spectral efficiency of 2 bit/s/Hz the optimistic maximum throughput can be derived to be 6.4 Mbps. In the first run, with a mutation probability of 5%, a maximum throughput of 3.2 Mbps was achieved, which had 16-QAM as the selected modulation scheme in its solution space. By increasing the probability to 25% from the set 2, the algorithm acquired a maximum throughput of 4.3 Mbps. In this case, V29 was selected as the modulation scheme to achieve such high throughput. Even in the previous run of 5% mutation probability, V29 was tried among its individuals, but the right combination of gain values was probably not selected.

The last run had a 50% mutation probability, which resulted in 4.67Mbps of throughput. Even this run selected V29 as its modulation scheme solution, but by attempting different values of gain, it was able to achieve such higher results. Increasing the mutation probability beyond 50%, did not result in an increase in throughput anymore. We also observe that in all three cases, a very good throughput result was attained in less than 3 generations. After which the best throughput of the each generation increases slowly. In the first case, the maximum throughput was attained in 7th generation, after which it saturated. For the remaining two cases, it was attained midway, after around 18th generation. Once the algorithm figures out the best solution, it configures the selected radio parameters of the SDR. By initiating a data transfer, through *nuttcp*, we consistently get the best throughputs achieved by the algorithm. Both the experiments were run for 30 generations each, which is certainly not enough to guarantee the global optimum. It usually only achieves a local optimum. But genetic algorithms are well known to converge to a global optima eventually [11] [7].

VI. Conclusion

In this paper, we presented a multi-objective genetic algorithm based optimization approach to configure SDRs. By giving a complete unknown radio environment with a wide range of input parameters, the algorithm optimizes the radio configuration of the SDR to result in an optimum wireless transmission, while achieving the objectives of the evaluation function. In our experimental system, we observed a 180% increase in throughput while using the proposed algorithm when compared to a known manual configuration of the SDR. Although the time taken to achieve the optimized solutions is not always feasible for the bootstrapping time of a radio, the algorithm proves efficient for radios that are non-mobile and where the wireless conditions do not change too much. Understandably, if the surroundings change to worsen the selected configuration, the algorithm needs to be restarted again.

The proposed algorithm improves the performance of the wireless link between any two SDRs by optimizing the radio parameters of frequency, bandwidth, modulation, coding, or any configurable parameter on which the wireless link depends on. It is certainly not possible for the user to manually try the huge solution space of radio configuration or even to understand how the combination of these parameters influence the wireless link. Moreover, the algorithm optimizes the radio properties of both the transmitter and the receiver at the same time. We also presented ideas to improve the efficiency of the optimization results and to increase the convergence speed to determine an optimal parameter set.

The current algorithm works, but can still be made better to fit any user requirement specifications, by optimizing the GA parameters, the selection of fitness functions and their weights, the population selection strategy, and the termination criteria. Like EVM from the physical layer, which is considered as one of the fitness functions in our experiments, other parameters from the MAC/PHY layer can be added to the algorithm to achieve the desired results. The change in mutation probability is done manually in each experiment to attain the desired results. In the future, we plan to change the mutation/crossover rate from generation to generation through the algorithm, starting from a higher rate and decreasing it over the evolution. This would allow the algorithm to try varied solutions before converging towards a local optimum. We further plan to incorporate the optimization framework into GNU Radio [14].

The source code of the entire prototype [15], which requires a modified Iris module [16] is available under an open-source license for further extensions and improvements.

REFERENCES

[1] C. Doerr, D. C. Sicker, and D. Grunwald, "Experiences implementing cognitive radio control algorithms," in *IEEE Global Telecommunications Conference (GLOBECOM)*, pp. 4045–4050, Nov. 2007.

[2] "Distributed Evolutionary Algorithms in Python (DEAP)." Available under https://github.com/DEAP.

[3] P. D. Sutton, J. Lotze, H. Lahlou, S. A. Fahmy, K. E. Nolan, B. Özgül, T. W. Rondeau, J. Noguera, and L. E. Doyle, "Iris: An Architecture for Cognitive Radio Networking Testbeds," vol. 48, pp. 114–122, Sept. 2010.

[4] A. Puschmann, "Open Software Platform for Experimental Cognitive Radio Research (OSPECORR)." Available under https://github.com/andrepuschmann/OSPECORR.

[5] A. Puschmann and Z. Shaik, "Building Self-Optimizing Radios using DEAP." Available under https://fosdem.org/2016/schedule/event/deap/.

[6] "An Overview on Genetic Algorithms." Available under http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html.

[7] T. Murata and H. Ishibuchi, "MOGA: Multi-Objective Genetic Algorithms," in *IEEE International Conference on Evolutionary Computation*, Nov. 1995.

[8] D. Kozel, "Optimization of digital modulation schemes using evolutionary algorithms.," *GNU Radio Conference*, 2015.

[9] "Pareto efficiency." https://en.wikipedia.org/wiki/Pareto_efficiency.

[10] C. J. Rieser, T. W. Rondeau, C. W. Bostian, and T. M. Gallagher, "Cognitive radio testbed: further details and testing of a distributed genetic algorithm based cognitive engine for programmable radios," in *IEEE Military Communications Conference (MILCOM)*, vol. 3, pp. 1437–1443, Oct. 2004.

[11] T. W. Rondeau, B. Le, C. J. Rieser, and C. W. Bostian, ""cognitive radios with genetic algorithms: Intelligent control of software defined radios"," *Proceeding of the SDR Technical Conference and Product Exposition*, pp. 93–100, 2004.

[12] T. W. Rondeau, *Application of Artificial Intelligence to Wireless Communications*. PhD thesis, Virginia Polytechnic Institute and State University, 2007.

[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, Apr 2002.

[14] "GNU Radio: The Free and Open Software Radio Ecosystem." Available under http://gnuradio.org/redmine/projects/gnuradio/wiki.

[15] "Self-Optimization of Software Defined Radios Through Evolutionary Algorithms." Available under https://github.com/andrepuschmann/OSPECORR/tree/optimizer_rebased.

[16] "IRIS Module for Self-Optimizer." Available under https://github.com/andrepuschmann/iris_modules/commits/optimizer.