A Nested Genetic Algorithm for Mobile Ad-Hoc Network Optimization with Fuzzy Fitness

NourElDin S. Eissa[1], Ahmed Zakaria Talha[2], Ahmed F. Amin[3], Amr Badr[4]
Department of Computer Engineering
Arab Academy for Science, Technology and Maritime Transport (AASTMT), Cairo, Egypt[1, 2, 3]
Department of Computer Science, Cairo University, Cairo, Egypt[4]

***Abstract*—One of the major culprits that faces Mobile Ad-hoc networks (MANET) is broadcasting, which constitutes a very important part of the infrastructure of such networks. This paper presents a nested genetic algorithm (GA) technique with fuzzy logic-based fitness that optimizes the broadcasting capability of such networks. While normally the optimization of broadcasting is considered as a multi-objective problem with various output parameters that require tuning, the proposed system taps another approach that focuses on a single output parameter, which is the network reachability time. This is the time required for the data to reach a certain percentage of connected clients in the network. The time is optimized by tuning different decision parameters of the Delayed Flooding with Cumulative Neighborhood (DFCN) broadcasting protocol. The proposed system is developed and simulated with the help of the Madhoc network simulator and is applied on different realistic real-life scenarios. The results reveal that the reachability time responds well to the suggested system and shows that each scenario responds differently to the tuning of decision parameters.**

Keywords*—Broadcasting; DFCN; fuzzy logic; genetic algorithms; Madhoc simulator; MANET

I. INTRODUCTION

Mobile Ad-hoc Networks (MANETs) are dynamic types of network consisting of an uncontrolled setup of end-point communication devices known as terminals, which are able of arbitrarily connecting with each other without the need of a base station or a fixed infrastructure [1]. The types of devices that are usually found in MANETs are laptops and smartphones equipped with limited range wireless technologies such as Bluetooth and WiFi (802.11). This, in turn, limits the communication capability of such devices, but allows them to move while communicating.

This makes the MANET very unpredictable as it needs to continuously self-reconfigure itself to accommodate these dynamic changes [2]. This is considered a major drawback for the efficiency and effectiveness of the MANETs and, by failing to readjust, link breakage will start to take place and some of the routes can become undiscoverable [3]. For the devices to be able to reach a certain destination, they start sending route discovery requests to their neighboring nodes [4] which, in turn, do the same thing. This results in the network being overwhelmed with an extreme amount of broadcast traffic known as a broadcasting storm [5].

Since it is clear that broadcasting plays a very critical role in network discovery and assists the nodes in MANETs in discovering their neighborhood [6], optimizing it constitutes a major step as it will save both energy and time, especially since most of the devices in the network have limited energy as they are battery powered.

Due to the previously mentioned limitations, a key threat known as node 'selfish behavior' arises in the network, in which the nodes purposely tend to drop the messages that do not target it, in an effort to save its energy [7] [8]. In other words, the nodes are not encouraged to contribute to the forwarding process. This kind of self-regarding behavior negatively impacts the network because, as already stated, there is no solid infrastructure in MANET and all the nodes rely on the cooperation of other nodes in the network to deliver and forward their messages. Delayed Flooding with Cumulative Neighbors (DFCN) is a broadcasting protocol that can handle this behavior and, at the same time, can reduce the number of packets that need forwarding with minimal punitive actions on the final coverage [9]. This is achieved by dropping the forwarded message when enough of the neighborhood devices have already got it. Also, once a node decides to forward a certain packet, it waits for a specified amount of time before executing this action, which is then canceled if another node in the network actually forwards the message [10].

The work proposed in this paper tackles a specific type of MANET, known as Metropolitan Mobile Ad-Hoc Networks, which is characterized by a disparate density that is continuously changing, whereas highly dense areas can swing from being active to inactive over short periods of time. Because creating a real testbed for this type of network is very costly and challenging, and might also lack the reproducibility factor, it was decided that the best approach to handle it is by means of a simulation framework. The Madhoc [11] simulator has been selected to achieve this. An evolutionary algorithm-based technique that combines nested GA with fuzzy-based fitness is proposed and implemented. The technique integrates the Madhoc simulator in its core and considers DFCN optimization over multiple real life mobility scenarios.

The rest of this paper is organized as follows. Section II introduces the Madhoc simulator and gives an insight about its capabilities and the different modes of operations. In Section III, a review of the related work concerning the optimization of broadcasting techniques in MANETs is presented. Section IV highlights the main problems that this research aims to solve. Section V demonstrates the algorithms

and techniques used to solve the problem. Section VI shows the obtained results and discusses them. Finally, Section VII concludes this work and proposes the potential future work.

II. MADHOC SIMULATOR

Madhoc is a metropolitan MANET simulator completely written in Java and available to use publicly [12] on the author's website [13]. The simulator provides the ability to simulate MANET using different parameters and real-life constraints such as working area size, mobility speed, wall thickness, etc. It also supports many different wireless technologies (e.g. WiFi, Bluetooth, GSM, etc.). Most importantly, it implements the full DFCN broadcasting protocol with all the required decision parameters to optimize it. Madhoc can be executed as a standalone application or as an Application Programming Interface (API).

To be able to collect the required statistics and results, a Madhoc monitor class is used. A monitor is not a part of the physical network and does not have an instance in real networks, and is regarded to as an abstraction entity that only exists at simulation level. It mainly aims at maintaining a global perspective on all nodes and for carrying out the required operations such as node deployment and initialization. It mainly serves as an observer of the Ad-hoc decentralized process. Another major attribute of the Madhoc simulator is that it does not use an event-driven simulation architecture, but instead, the simulator's kernel iterates upon a discrete time domain, where the distance between two intervals is known as the resolution.

This parameter is defined by the user and should be fixed throughout all the related applications to guarantee comparable and consistent results. The higher this value is, the less accurate the simulation will become. This value should be carefully used according to the required application. In the case of DFCN, this value must be at least twice lower than the maximum RAD, otherwise the benefits of using RAD will be completely lost.

Another important factor to consider while choosing the resolution is the mobility scheme of the nodes, the resolution must be small enough to make sure that the nodes move in reasonable steps, otherwise, some connections that could have taken place in real life would not be simulated.

III. RELATED WORK

In the literature, most research has been dedicated to solving the broadcasting issues by using a multitude of different methods. Evolutionary multi-objective approaches have been proven to be effective in solving broadcasting problems [14], however, they suffer from time and performance issues [15]. Other methods focus on combinatory numerical models but most of them fail to adequately reduce the routing overhead with highly scalable networks, which is a main feature of MANET. Those who focused on the DFCN protocol did not formulate a trending mobility model for optimizing the decision parameters. Some of the researchers directly focused on detecting the selfish nodes in the network and avoiding them to increase the efficiency of the broadcasting protocols, the most notable work in this regard is by S. Subramaniyan et al. [16], where a Record-and-Trust-Based Detection (RTBD) technique was simulated that can efficiently detect selfish nodes in MANET. The main focus of this work was to accelerate the detection of misbehaving selfish nodes. The proposed method managed to diminish the overhead, latency and overhead ratio which improved the broadcasting performance of the MANET. However, the authors did not demonstrate how the acquired security could be transferred to the neighboring nodes in the network so that they could avoid being compromised by the selfish nodes detected by RTBD, meaning that the technique is not scalable on larger networks and the performance will be degraded. Another key focus in the literature is intelligent rebroadcasting techniques that reduce the overhead by estimating the usefulness of rebroadcasts and the probability of causing a collision. S. S. Basurra et al. [17] discussed a Zone based Routing with Parallel Collision Guided Broadcasting Protocol (ZCG) to reduce redundant broadcasting and to accelerate the path discovery process. The authors compared ZCG with two other techniques, Dynamic Source Routing (DSR) and Adhoc Ondemand Distance Vector Routing (AODV). It was concluded that ZCG can speed up the routing process in MANET due to its on-demand parallel collision guided broadcasting. However, the proposed method lacked distribution fairness among the nodes and did not protect zone members from selfish behavior attributed to the Zone Leader. Another interesting finding in the literature is the clustering of MANETs as a mean to reduce the complexity of the routing table. M. Ahmad et al. [18] provided a comprehensive survey about the different clustering algorithms that address this issue. It concluded that the effectiveness of the clustering algorithms depends on a set of specific parameters, which the nodes are remaining power, the relative mobility, the overhead data, the trust value, and the node reputation.

IV. PROBLEM STATEMENT

In order to optimize the DFCN protocol, multiple decision parameters need to be considered. These parameters dictate how DFCN operates and they characterize the search space. Since the optimization heavily relies on each specific scenario, an individual optimization trend is expected for each scenario.

The reachability time t_r is the output benchmark that is used to measure the optimization result. It is the amount of time required for the network to reach a certain number of pre-defined nodes. The goal of this research is to optimize the DFCN parameters to decrease the reachability time of the nodes inside the MANET. The problem is formulated as follows:

$$m: instance\ of\ Madhoc\ simulator, t_r: reachability\ time.$$

$$t_r = m(LowerRAD, UpperRAD, ProD, MinGain, SafeDensity) \quad (1)$$

$$f(LowerRAD, UpperRAD, ProD, MinGain, SafeDensity) = min\ (t_r)$$

The function f corresponds to the proposed system where the target is to minimize the reachability time t_r for each instance of the simulator *m.* Table I below shows the DFCN parameters along with their respective threshold and domain values.

TABLE. I. DFCN PARAMETER DESCRIPTION

Parameter Name	Domain	Description	Unit	Threshold Value
LowerRAD	Real (ℝ)	Minimum time required to rebroadcast.	Second	[0, UpperRAD]
UpperRAD	Real (ℝ)	Maximum time required to rebroadcast	Second	[LowerRAD, 10]
ProD	Integer (ℤ)	Maximum Density for which it is still required to use proactive behavior (reacting to new neighbors)	Device	[0, 100]
MinGain	Real (ℝ)	Minimum gain for rebroadcasting.	-	[0, 1]
SafeDensity	Integer (ℤ)	Maximum density, below which the protocol will always broadcast.	Device	[0, 100]

As already stated, this will be done on three different mobility model scenarios, namely, Highway, Mall and Human mobility. The description for these scenarios is shown next.

A. Highway Scenario

The main feature of the highway mobility model is that the nodes move at significantly higher speeds compared to the other mobility models and the nodes are lower in numbers. The spot density is also set to one spot per square kilometer, which is very sparse, and the number of spots per simulation area is limited to three. In this scenario, most of the generated traffic comes from nodes moving in opposite directions to simulate cars moving on different and opposing lanes of a highway. Table II below shows the properties of this scenario.

B. Mall Scenario

The mall mobility scenario is composed of separate regions connected by relatively narrow areas. It represents a group of shops interconnected using corridors. In this scenario, the surface area is smaller than the highway one and the velocity is much slower. Also, the nodes move randomly for most of the time with no clear targets, representing humans wandering around and shopping in arbitrary shops. Table III illustrates the different parameters for this scenario.

C. Human Mobility Scenario

This scenario is more distinctive than the mall one and is considered one of the most daunting models. In this context, the focus is on the human mobility scheme, where the movements are not random, but instead, there is a list of target destinations that each node mostly moves towards. These targets can be far away, as well as a few meters around. Also, the targets can dynamically change with time depending on human behavior. For instance, a waiter in a restaurant can be regularly moving back and forth between the kitchen and customers' tables.

TABLE. II. HIGHWAY MOBILITY SCENARIO PARAMETERS

Parameter	Value	Units
Surface Area	1 x 1	km^2
Nodes Density	80	nodes / km^2
Velocity	[20 40]	$m.s^{-1}$

TABLE. III. MALL MOBILITY SCENARIO PARAMETERS

Parameter	Value	Units
Surface Area	0.3 x 0.3	km^2
Nodes Density	6,500	nodes / km^2
Velocity	[0.3 1]	$m.s^{-1}$

TABLE. IV. HUMAN MOBILITY SCENARIO PARAMETERS

Parameter	Value	Units
Surface Area	0.05 x 0.05	km^2
Nodes Density	80,500	nodes / km^2
Velocity	[0.3 1.5]	m/s^{-1}

The human mobility scheme is defined as a round simulation area, where fixed places that act as target spots are scattered and where the distance between two places cannot be less than 10 meters. Table IV shows the parameters for the human mobility model.

V. PROPOSED SYSTEM

The proposed technique consists of nested GA with fuzzy-based fitness. The aim is to optimize the DFCN decision parameters according to the reachability time and to find certain trends for each one of the different scenarios. The benchmark used is the reachability time for 10% of the nodes, which is the time required so that 10% of the nodes in the network successfully deliver their messages. The outer GA contains the DFCN parameters and the to-be-calculated output from the simulator. The inner GA evolves a set of rules for the fuzzy system, where each chromosome represents a complete fuzzy set and the inference output represents the inner fitness. The final inner fitness value that is calculated after the convergence has completed sets the fitness value of the outer GA. The proposed system is developed using C# language on Microsoft Visual Studio 2017 under 64-bit Windows 10 with 8GB of RAM and an Intel Core i5-6500 CPU. Because the proposed system is built using C# and the Madhoc simulator operates fully in Java, a mechanism that interfaces them was required. To be able to accomplish this, each time the simulator is required to calculate the reachability time, it is executed by the developed application as a command line program running inside a virtual sandbox process, where all the standard inputs and outputs are redirected to the application. Fig. 1 shows an overview of the system.

Fig. 2 shows the pseudo-code of the proposed system.

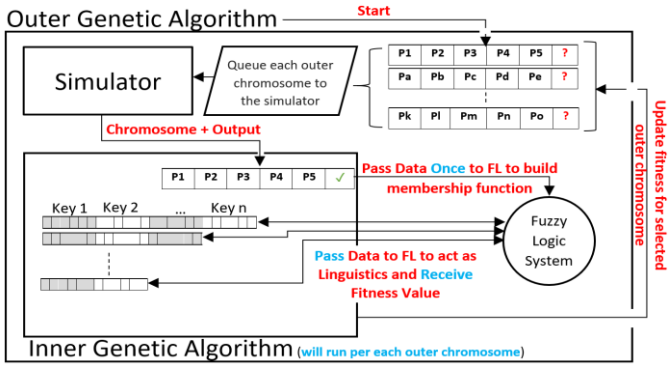

Fig. 1. Proposed System Illustration.

```
1  FUNCTION RunInnerGA(innerGenerationsCount, oChromosome)
2      i ← 0;
3      Pi ← InitializeInnerPopulation(innerPopulationSize, keyMin, keyMax );
4      WHILE ( i < innerGenerationsCount -1 )
5          FL ← BuildFuzzySystem_VariableSets( oChromosome );
6          Pi+1 ← NULL;
7          j ← 0;
8          WHILE ( j < innerPopulationSize)
9              FL ← InitializeFuzzySystemLinguistics( Pi[ j ] );
10             Fitness( Pi[ j ] ) ← FuzzySystem_InferenceResult( );
11             j ← j + 1;
12         END WHILE
13         j ← 0;
14         WHILE ( j < innerPopulationSize / 2 - 1)
15             parents = RouletteSelect( Pi );
16             offspring[0,1] = Crossover(parents, innerGACrossoverProbability);
17             Pi+1 ← Pi+1 + offspring[0,1 ];
18         END WHILE
19         j ← 0;
20         WHILE ( j < innerPopulationSize)
21             Pi[ j ] = Mutate( Pi[j ], innerGAMutationProbability);
22             j ← j + 1;
23         END WHILE
24         Pi+1 ← Pi+1 + GetFittest( Pi );
25         i ← i + 1;
26     END WHILE
27     RETURN GetHighestFitnessValue( Pi );
28 END FUNCTION

30 FUNCTION RunOuterGA(outerGenerationsCount): MAIN
31     i ← 0;
32     Pi ← InitializePopulation(outerGenerationsCount, thresholdsValues[ ]);
33     WHILE ( i < outerGenerationsCount -1 )
34         Pi+1 ← NULL;
35         j ← 0;
36         WHILE( j < outerPopulationSize)
37             Output( Pi[ j ]) ← GetMadhocOutput( Pi[ j ]);
39             Fitness( Pi[ j ] ) ← RunInnerGA ( Pi[ j ] );
40             j ← j+1;
41         END WHILE
42         j ← 0;
43         WHILE( j < OP_ outerPopulationSize / 2 - 1)
44             parents = RouletteSelect(Pi) ;
45             offspring[0,1] = Crossover(parents, outerGACrossoverProbability);
46             Pi+1 ← Pi+1 + offspring[0,1];
47         END WHILE
48         j ← 0;
49         WHILE ( j < outerPopulationSize)
50             Pi[ j ] = Mutate( Pi[ j ], outerGAMutationProbability);
51             j ← j + 1;
52         END WHILE
53         Pi+1 ← Pi+1 + GetFittest( Pi );
54         i ← i + 1;
55         ExtractParametersAndOutput( Pi );
56     END WHILE
57 END FUNCTION
```

Fig. 2. Pseudo-Code for the Proposed System.

The RunOuterGA function is the entry point of the program. The InitializeInnerPopulation function creates the initial population with random fuzzy logic keys that correspond to the linguistic strings. The oChromosome variable is the outer chromosome passed from the outer GA to the inner one, per generation. The keyMin and keyMax variables represent the range for the allowed number of keys per chromosome. At line 5, the fuzzy logic system is initialized and the fuzzy sets are created using the oChromsome genes, then at line 9, the linguistics are generated using the inner chromosome Pi[j], and finally at line 10, the fitness is calculated by getting the inference result for the developed fuzzy logic system.

The ExtractParametersAndOutput function is called per each outer GA generation to extract the current values of the decision parameters and the output from the fittest chromosome.

A. The Fuzzy Logic System

The fuzzy system is used to calculate the fitness for the inner GA. Each chromosome from the inner GA will act as complete fuzzy set. Each DFCN parameter will act as a linguistic variable with LOW, MED and HIGH as values. All of the variables have a triangular membership function that is equally divided over the maximum threshold of the respective parameters it represents. The rules for the fuzzy set are generated and optimized using the inner GA, which will be highlighted later.

In order to accomplish this, the inner chromosome is decoded from a numerical form to equivalent linguistic strings, according to Table V. To get the output values, the inference system uses a centroid defuzzifier with an interval of 1000. The interval represents the number of segments that the linguistic universe will be split into to perform the numerical approximation of the area center.

B. Outer Genetic Algorithm

The chromosome structure for the outer GA contains a hybrid of floating-point and integer values that correspond to the DFCN parameters, and also contain the output parameter which corresponds to the reachability time that will be calculated using the Madhoc simulator.

The chromosome size for the outer GA has a fixed length of six genes. Fig. 3 illustrates the chromosome structure. The crossover is a standard single-point operator that takes into consideration the gene placement to make sure the swapped parameters are still compatible and are within the specified thresholds. The mutation is performed through a non-uniform operator, which can be used to limit the lower and upper boundaries for the genes - which is crucial to avoid out-of-boundaries parameters - and also because it prevents the population from stagnating during the early evolution stages. The outer population size is fixed at 100 chromosomes and runs for a maximum of 300 generations. The crossover and mutation probabilities are fixed at 30% and 10%, respectively.

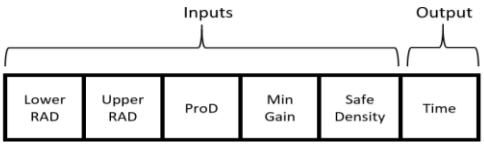

Fig. 3. Outer Chromosome Structure (Size=6).

TABLE. V. NUMERICAL-TO-LINGUISTIC STRING CONVERSION TABLE

Value	Equivalent Linguistic
1	LOW
2	MED
3	HIGH
-1	NOT LOW
-2	NOT MED
-3	NOT HIGH
0	NOT APPLICABLE

The selection is done through a traditional Roulette-Wheel operator. It is worth noting that the last gene (reachability time) is excluded from the evolution process and is stored inside the chromosome and passed later to the fuzzy system. All of the other aforementioned decision parameters are randomly generated within the threshold.

C. Inner Genetic Algorithm

The inner GA uses the same operators as the outer one. However, the chromosome structure is different. It consists of a variable number of genes ranging from 3 to 15. Each gene represents a key that encodes a linguistic string into numerical values as shown previously. This had to be done in order to be able to evolve the rules using the GA. Each key has a fixed length of 6 which corresponds to the number of input parameters and the output parameter.

The population size for the inner GA is set to 50 and the maximum number of generations is 100. Fig. 4 illustrates a sample inner GA with a population size of 7 and random chromosome sizes, denoted with S_n, where n is the chromosome number inside the population. It also shows an example of how the key is decoded into a linguistic string. The inner GA makes a complete run of 50 generations for each outer chromosome. The target is to diversify the linguistics of the fuzzy logic to reach the best possible output.

The defuzzified output value represents the fitness of the outer chromosome. After doing this for all the outer GA chromosomes, the best one is chosen and the outer GA transits into the next generation.

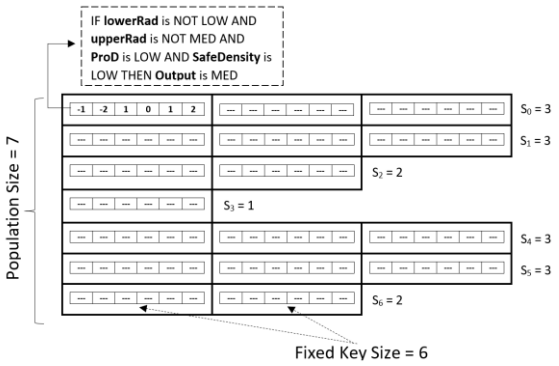

Fig. 4. Inner Genetic Algorithm Illustration.

VI. RESULTS AND DISCUSSION

The experiments are run five times and the results are averaged. The results show the convergence of decision parameters and the output (solid black line). The logarithmic trendline (red dotted line) is also calculated to provide a mathematical model for the decision parameters. Fig. 5 shows the results for the highway mobility environment. Table VI shows the output trendline for each decision parameter and the equivalent logarithmic regression expressions.

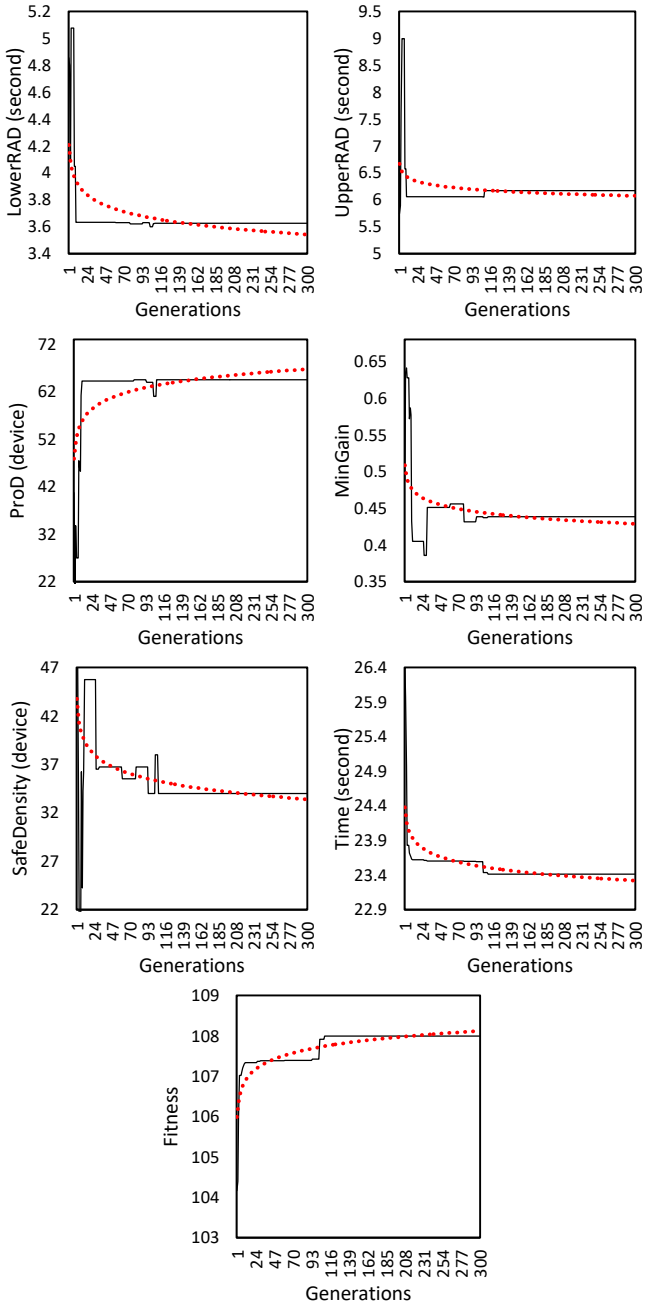

Fig. 5. Convergence for the High Way Mobility Model.

TABLE. VI. TRENDLINE PARAMETERS FOR HIGHWAY SCENARIO

Parameter	Trendline	Expression
LowerRAD	↓	$-0.117 * ln(G) + 4.2076$
UpperRAD	↓	$-0.105 * ln(G) + 6.6723$
ProD	↑	$3.3079 * ln(G) + 47.885$
MinGain	↓	$-0.014 * ln(G) + 0.5087$
SafeDensity	↓	$-1.818 * ln(G) + 43.76$

Fig. 6 shows the results for the Mall mobility scenario and Table VII shows the trendline for the decision parameters. The results for the human mobility model are shown in Fig. 7 and the respective trendline parameters are shown in Table VIII.

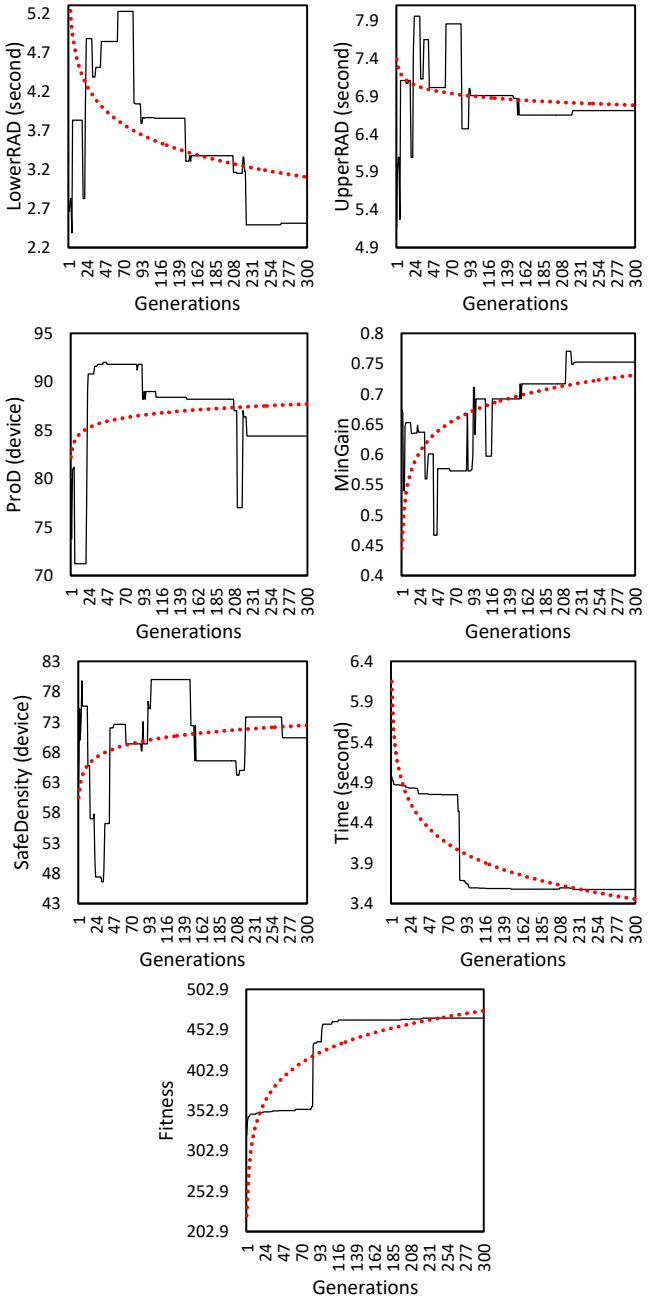

Fig. 6. Convergence for the Mall Mobility Model.

TABLE. VII. TRENDLINE PARAMETERS FOR MALL SCENARIO

Parameter	Trendline	Expression
LowerRAD	↓	$-0.463 * ln(x) + 5.7409$
UpperRAD	↓	$-0.106 * ln(x) + 7.3855$
ProD	↑	$0.9608 * ln(x) + 82.223$
MinGain	↑	$0.0502 * ln(x) + 0.445$
SafeDensity	↑	$2.0984 * ln(x) + 60.488$

In the highway mobility scenario, the time to reach the destination decreased from 26.44 to 23.41 seconds, which amounts to 11.45%. Given that the number of nodes in this network is 80, the average time for a node to deliver a message decreased from 3.3 to 2.92 seconds.

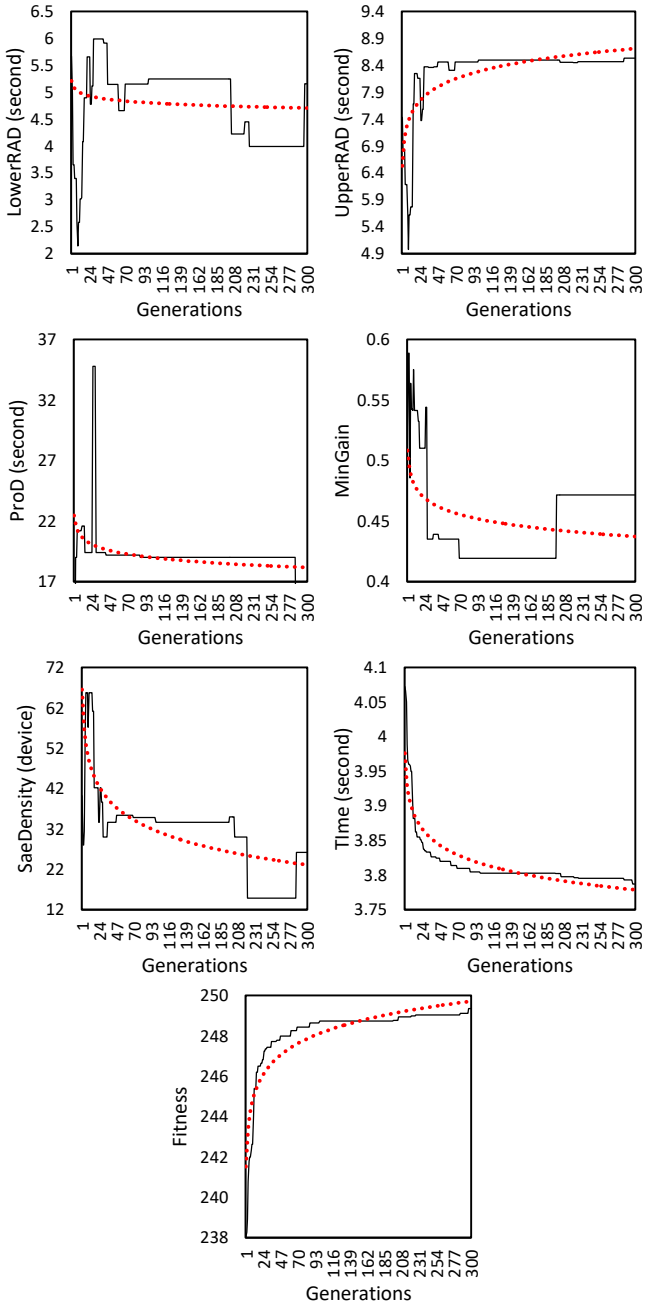

Fig. 7. Convergence for the Human Mobility Model.

TABLE. VIII. TRENDLINE PARAMETERS FOR HUMAN MOBILITY SCENARIO

Parameter	Trendline	Expression
LowerRAD	↓	$-0.088 * ln(x) + 5.2098$
UpperRAD	↑	$0.3832 * ln(x) + 6.526$
ProD	↓	$-0.757 * ln(x) + 22.479$
MinGain	↓	$-0.012 * ln(x) + 0.5084$
SafeDensity	↓	$-7.621 * ln(x) + 66.544$

For the mall mobility scenario, the time to reach the nodes decreased from 4.98 seconds to 3.57 seconds which amounts to 28.3%, which brings down the average required time to deliver a message from 7.6ms to 5.49ms.

As for the human mobility scenario, the time to deliver the messages to their respective destinations decreased from 4.07 to 3.78 seconds, which amounts to 7.12%. The average time to deliver a message decreased from 0.5ms to 0.46ms.

By inspecting all the previous results, it appears that the mall mobility model benefited the most from the optimization of the DFCN decision parameters and the human mobility model benefited the least. While these two models have very close features, the major difference between them, as stated previously, is the randomness of the movements. The human mobility model is governed by human intentions of moving between a dynamic list of targets while the mall one is governed by random motion of shoppers moving between random shops. Also, by inspecting the highway scenario, it seems that the lack of enough nodes has significantly raised the average delivery time six times (6x) the delivery time in other scenarios.

To demonstrate the consistency of the results, a 5% confidence interval for the final reachability time is calculated and is shown in Table IX.

TABLE. IX. 5% CONFIDENCE INTERVAL FOR THE FINAL REACHABILITY TIME

Mobility Model	5% Confidence Interval (seconds)
Highway	23.4 ± 0.75
Mall	3.57 ± 0.34
Human	3.78 ± 0.02

VII. CONCLUSION AND FUTURE WORK

The proposed system managed to decrease the message delivery time for the three real-life scenarios (the highway, the mall and the human mobility models) by optimizing the decision parameters for the DFCN protocol. The mall mobility model benefited the most from the optimization of the DFCN parameters, which is mainly attributed to the randomness of the mobility, since the human mobility model also shares very close parameters but only differs in the movement intention. In the human mobility model, the mobility is governed by the intentions of the humans to reach a certain dynamic list of destinations and, therefore, the randomness significantly decreases. Also, the highway mobility model yielded the highest average message delivery time, which is attributed to the lack of nodes and the very high mobility speed, and since the DFCN protocol relies on 1-hop neighbors to deliver the messages to their destinations, this scenario severely affects it.

In the future, a mathematical model based on the found trendlines can be established and tested. This will help to achieve the results faster, instead of relying solely on metaheuristic techniques, which require a significant amount of time to converge to the optimal solution.

Also, Genetic Programming (GP) can be experimented with, to evolve programs and expressions related to each scenario. This way, the resulting programs can be used as a rigid optimization model, without the need to repeat the evolution process each time.

REFERENCES

[1] V. Rishiwal, S. K. Agarwal and M. Yadav, "Performance of AODV protocol for H-MANETs," in International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Spring), Dehradun, India, 2016.

[2] L. J. G. Villalba, J. G. Matesanz, A. L. S. Orozco and J. D. M. Díaz, "Auto-Configuration Protocols in Mobile Ad Hoc Networks," Sensors (Basel), vol. 11, no. 4, p. 3652–3666, 2011.

[3] C. Dhakad and A. S. Bisen, "Efficient route selection by using link failure factor in MANET," in International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, India, 2016.

[4] P.-J. Chuang, P.-H. Yen and T.-Y. Chu, "Efficient Route Discovery and Repair in Mobile Ad-hoc Networks," in IEEE 26th International Conference on Advanced Information Networking and Applications, Fukuoka, Japan, 2012.

[5] V. Sharma and A. Vij, "Broadcasting methods in mobile ad-hoc networks," in 2017 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, 2017.

[6] M. Bakhouya, "Broadcasting approaches for Mobile Ad hoc Networks," in International Conference on High Performance Computing & Simulation (HPCS), Helsinki, Finland, 2013.

[7] H. Yadav and H. K. Pati, "A Survey on Selfish Node Detection in MANET," in International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), Greater Noida (UP), India, 2018.

[8] N. Ramya and S. Rathi, "Detection of selfish Nodes in MANET - a survey," in International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2016.

[9] L. Hogie, P. Bouvry, M. Seredynski and F. Guinand, "A Bandwidth-Efficient Broadcasting Protocol for Mobile Multi-hop Ad hoc Networks," in International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL), Morne, Mauritius, 2006.

[10] B. Dorronsoro, P. Ruiz, G. Danoy, Y. Pigné and P. Bouvry, "BROADCASTING PROTOCOL," in Evolutionary Algorithms for Mobile Ad Hoc Networks, John Wiley & Sons, Inc, 2014, pp. 135-138.

[11] L. Hogie and P. Bouvry, "An Overview of MANETs Simulation," Electronic Notes in Theoretical Computer Science, vol. 150, no. 1, pp. 81-101, 2006.

[12] L. Hogie, F. Guinand and P. Bouvry, The Madhoc metropolitan ad hoc network simulator, France: Luxembourg University and Le Havre University, 2006.

[13] L. Hogie, "Madhoc Metropolitan ad hoc network simulator," 2006. [Online]. Available: http://www.i3s.unice.fr/~hogie/madhoc/. [Accessed 25 January 2019].

[14] R. M. Chintalapalli and V. R. Ananthula, "M-LionWhale: multi-objective optimisation model for secure routing in mobilead-hocnetwork," IET Communications, vol. 12, no. 12, pp. 1406 - 1415, 2018.

[15] E. Alba, B. Dorronsoro, F. Luna and P. Bouvry, "A cellular multi-objective genetic algorithm for optimal broadcasting strategy in metropolitan MANETs," in IEEE International Parallel and Distributed Processing Symposium, Denver, CO, USA, 2005.

[16] S. Subramaniyan, W. Johnson and K. Subramaniyan, "A distributed framework for detecting selfish nodes in MANET using Record- and Trust-Based Detection (RTBD) technique," EURASIP Journal on Wireless Communications and Networkingvolume, p. Article 205, 2014.

[17] S. S. Basurra, M. D. Vos, J. Padget, Y. Ji, T. Lewis and S. Armou, "Energy Efficient Zone based Routing Protocol for MANETs," Ad Hoc Networks, vol. 25, pp. 16-37, 2015.

[18] M. Ahmad, A. Hameed, A. A. Ikram and I. Wahid, "State-of-the-Art Clustering Schemes in Mobile Ad Hoc Networks: Objectives, Challenges, and Future Directions," IEEE ACCESS, vol. 7, pp. 17067 - 17081, 2019.