Optimal Topology Generation for Linear Wireless Sensor Networks based on Genetic Algorithm

Adil A. Sheikh\textsuperscript{1}
Science and Technology Unit
Umm Al-Qura University
Makkah, Saudi Arabia

Emad Felemban\textsuperscript{2}
College of Computing and Information Systems
Umm Al-Qura University
Makkah, Saudi Arabia

Abstract—A linear network is a type of wireless sensor network in which sparse nodes are deployed along a virtual line; for example, on streetlights or columns of a bridge, tunnel, and pipelines. The typical deployment of Linear Wireless Sensor Network (LWSN) creates an energy hole around the sink node since nodes near the sink nodes deplete their energy faster than others. Optimal network topology is one of the key factors that can help improve LWSN performance and lifetime. Finding optimal topology becomes tough in large network where total possible combinations is very high. We propose an Optimal Topology Generation (OpToGen) framework based on genetic algorithm for LWSN. Network deployment tools can use OpToGen to configure and deploy LWSNs. Through a discrete event simulator, we demonstrate that the use of genetic algorithm accomplishes fast convergence to optimal topologies as well as less computational overhead as compared to brute force search for optimal topology. We have evaluated OpToGen on the number of generations it took to achieve the best topology for various sized LWSNs. The trade-off between energy consumption and different network sizes is also reported.

Keywords—Ad hoc networks; network topology; genetic algorithms; computer simulation; computer networks management; network lifetime estimation; optimization

I. INTRODUCTION

Linear Wireless Sensor Network (LWSN) is a special kind of wireless sensor networks where nodes are placed in a linear formation (Fig. 1). LWSNs are used in applications involves monitoring or collecting data from infrastructure that spans over long distances such as pipelines, highways, bridges, and borders. Usually, WSN is deployed in 2D or 3D form where nodes are connected with multiple nodes in all directions. LWSN, by definition, on the other hand, have minimal connectivity. Fig. 2, shows typical connectivity scenarios for a linear network.

Typically, LWSN does not need sophisticated routing protocols since nodes generate packets and forward them to the next node closer to the sink node. As a result, nodes closer to the sink node have a higher load and thus deplete their batteries faster than others creating an energy hole around the sink node. There have been enormous efforts [1], [2], [3], [4] to characterize and study this kind of networks, including throughput improvement, lifetime enhancement, end-to-end delay minimization, and others. Topology configuration in a linear network is minimal; however, it is imperative since performance is much dependent on it [4].

Topology optimization in WSN has been an active research area recently [5] where selecting topology that satisfies specific criteria is the objective. Approaches such as Ant Colony Optimization, Swarm Optimization, Neural Networks, Genetic Algorithms, and Artificial Intelligence have been presented to solve the topology optimization problem [6]. Furthermore, the optimization of simple deployment like LWSNs would become complex when many parameters affect the outcome. The use of LWSN in smart cities’ intelligent transportation systems has pushed researchers to refocus on finding optimal topologies. Our proposed framework leverages the advancements in genetic algorithms to address complex problems, such as increasing network lifetime by optimizing network topology.

In our previous work [7], we introduced OpToGen concept. In this paper, we propose OpToGen framework that uses GA in conjunction with a network simulator to optimally deploy a heterogeneous LWSN. The optimization module that uses GA to maximize network lifetime is the main contribution of this paper. We constructed a fitness function based on network parameters related to network lifetime. Using GA has resulted in significantly reducing the computational overhead in finding the best or near-best network topology. We validate the practicality of OpToGen by implementing the framework on Matlab and NS-2. In future works, the researchers would...
like to address limitations of OpToGen, such as scalability.

The rest of the paper is divided into the following sections. The next section, Section II, discusses related research. Section III discusses OpToGen framework and its design. The implementation of OpToGen framework as a tool and its evaluation are presented in Section IV. The paper is summarized in Section V. Finally, possible future directions are presented in Section VI.

II. RELATED WORK

In the past decade, there has been a significant advancement in research on various issues related to LWSNs. Topological optimization of networks is of major importance due to its requirement in many domains, including telecommunications, electricity distribution, underwater cables, and oil/gas pipelines. Topology optimization can solve challenging problems like network lifetime maximization, node coverage, energy consumption, security, and reliability.

The author in [8] provides a comprehensive overview of linear wireless sensor networks (LWSNs) including its concept, applications and motivation to design specific network protocols that explores linearity of a network for energy savings, increasing lifetime, fault tolerance and reliability. A detailed hierarchical and topological classification of LWSN is also provided. A case study based on associated protocols is presented to increase the robustness and efficiency of routing for network optimization.

An efficient and reliable LWSN design is presented by [9]. In their design, the requirement of energy consumption and long network lifetime are met using an optimal node deployment scheme for data flow. Network efficiency is the primary optimization objective. The node deployment scheme optimizes the number of clusters and the number of relay and sensor nodes to be considered to form a linear network. Results of theoretical analysis and simulations for their work show that the method of nodes placement and topology control can solve energy consumption problems and improve network efficiency.

In [10], the author provides an analytical framework for node placement in a linear array fashion for uniform energy dissipation of all sensor nodes. This approach makes sure that all sensor nodes in the network die out simultaneously. The results show that the network lifetime is doubled as compared to other mentioned approaches. The author has also mentioned issues related to random node placement, as this theoretical requirement is not fulfilled in real scenarios. The random node placement with fixed bin length outperforms random node placement with variable bin length.

An efficient node placement scheme is presented in [11] to maximize the lifetime of LWSN through uniform energy conservation. The performance of uniform and random and linearly decreasing distance (LDD) node placement schemes are evaluated. The impact of gateway location on network performance is also analyzed. In a random node placement scheme, the nodes are placed randomly. For a connected network, it is assumed that each node is in communication range of the previous one. In uniform node placement scheme, the nodes are placed at equal distance with GW at the edge. In this case, nodes near GW lose energy soon due to additional data from remote sensors. In LDD, the distance between nodes is decreased toward GW to balance the load and increase the lifetime of the network. All these approaches result in increased network lifetime when GW is placed in the middle due to the reduced overhead of data forwarding in nearby nodes.

In literature, topological optimization problems can also be solved by heuristic methods viz. simulated annealing [12], Artificial Neural Network [13] and GA [14], [15] are preferred because of their strength and ability to find near-optimal solutions. Researchers have used GA to optimize different parameters of various network topologies. A survey is provided in [16] based on the application of GAs for sensor network optimization. The authors provide a comprehensive review modeling sensor communication in clustering and routing problems using GA. They also evaluate various GA-based optimization strategies.

In [17], the authors present an energy efficient network layout based on GA. The objectives taken into consideration are coverage and lifetime for network reliability. A novel method is introduced to decrease communication energy by clustering nodes and positioning them in the closest possible distance in each cluster. In their work, nodes are positioned in the network using GA and nodes have been divided into specific clusters using a K-means clustering algorithm. Another effort is reported by [18] where an algorithm is introduced to maximize the reliability of the network. In their algorithm, a different and efficient chromosome, as well as gene encoding and cross over, is used for better convergence towards the optimal solution. The population size and the computational time of the networks ensure that the proposed method converges to its optimal solution in very few CPU seconds. An efficient topology control using GA based cluster head selection is presented by [19].

In their clustering technique, the 3-tier sensor network architecture is developed comprising of super head nodes, cluster head (CH) nodes, and cluster slave nodes. This architecture utilizes node’s residual energy, bandwidth, and memory capacity to reduce the energy consumption of the network. Research work of [20] presents an algorithm to optimize the design of communication network topologies that are based on GA. A set of links for a given set of nodes is chosen to maximize reliability. Another complication that their algorithm solves is the calculation of computationally expensive reliability fitness function for promising candidate networks.

The study of related works mentioned above suggests that most research works have been carried out to optimize topologies for improving the reliability of WSNs. Some researchers have directed their research towards finding out optimal clustering of nodes to reduce communication overhead, gene and chromosome encoding to reduce computation time, node deployment scheme for energy efficiency and online interactive algorithm to maximize network lifetime. Others have provided ways to validate optimal topologies through various means. The work done in this paper tries to determine the best or near-best topology in lesser time and computation overhead using a novel GA under some constraints set on LWSNs.
In this research, we focus on two-tiered LWSN however, OpToGen generates optimal topologies for multi-tiered LWSNs. Fig. 3 depicts the scenario under discussion where sensor nodes or cameras have been attached to light poles along a highway. A sink node that collects and process data from sensor and camera node is placed in the network where it is in the direct transmission range of only a few nodes. Therefore, each sensor node relays its collected data through relay nodes to the sink node.

In this way, sensor nodes with limited memory, low processing power, and low battery power are part of tier-2 from the sink, whereas, the higher powered relay node is part of tier-1. Relay nodes are not limited to forwarding data from other nodes. They can also take part in sensing.

The scenario described above is given as input to OpToGen. OpToGen’s role as a black box is shown in Fig. 4. The number of nodes and other network parameters is given as input to OpToGen framework. Using GA and network simulator, OpToGen takes significantly less time to generate optimal or near-best topology. This is explained in Section III of the paper.

Finding optimal topology for a dense LWSN has been actively researched for some time now, but using network simulations to evaluate the fitness function is a novel idea. In [21], the researchers show a classic decision-making process based on GA. Fig. 5a explains their work. The first step is to create an initial population of solutions. The second step is to assess the fitness of that population. No further action is required if the desired fitness is achieved. Otherwise, the breeding process is carried out using genetic operators to create a new generation. Next, the new generation is evaluated. This evolution cycle continues until fitness no longer improves. We have modified the classic decision-making process of GA. As depicted in Fig. 5b, a network simulator is introduced for evaluating the current population’s fitness.

III. OpToGen Framework

Evolutionary Computation (EC) is the field of study that focuses on the simulation of natural evolution as a genetic solver for intractable problems and optimizations. OpToGen is based on EC. To achieve higher network lifetime, GA determines the optimal topology by determining optimal connections between LWSN nodes.

The working of OpToGen framework is explained using Algorithm 1. The inputs to OpToGen include N, the total number of sensor nodes to be deployed. Note that for a structure like a highway, the architect would know the total light poles on the highway. The remaining inputs are for GA. These include G, the maximum number of generations that the GA should run. Pc is the probability of crossover and Pm is the probability of mutation. The ratio of parents to offsprings in the new generation is R. Finally, I is the number of individuals in each population.

Like any other EC, OpToGen starts with generating its primary population. A random population of candidate topologies or candidate chromosomes is generated by Generate_Random_Population() at the start of the algorithm. The next sub-section explains how a chromosome represents a candidate topology.

A. OpToGen’s Genetic Algorithm

The first step to use GA for problem-solving is to design an appropriate encoding of the problem’s candidate solution. This design problem in itself is challenging. A better candidate solution representation as a chromosome is essential for the efficiency of GA. This step includes both genotypes and phenotypes for the GA.
Algorithm 1: Pseudocode for Generating Optimal Topology

```
1 Function OpToGen (G, N, I, R, P_C, P_M);
   Input : G is the maximum number of generations that the algorithm should run.
   I is the number of individuals in each generation.
   N is the number of nodes in the network.
   R is the ratio of offsprings chromosomes to parent chromosomes in the new generation.
   P_C and P_M are the probability of crossover and mutation, respectively.
   Output: Best_Topology is the optimal network topology and Best_Fitness its fitness.

2 G_1 ← Generate_Random_Population(N)
3 i ← 1;
4 Fitnesses ← ∅
5 repeat
6   Topology_i ← Translate_Chromosome(G_1(i))
7   Fitness_i ← Network_Simulator(Topology_i)
8   Fitnesses ← (Fitnesses ∪ Fitness_i)
9   i ← i + 1;
10 until i ≤ I;
11 g ← 1;
12 repeat
13   i ← 1;
14   Offsprings ← ∅
15   repeat
16       parent_1 ← Select_Random_Parent(G_g)
17       parent_2 ← Select_Random_Parent(G_g)
18       p_c ← Random_Number()
19       if p_c ≤ P_C then
20           offspring_i ← Crossover(parent_1, parent_2)
21       else
22           offspring_i ← Select_Random_Parent(parent_1, parent_2)
23       p_m ← Random_Number()
24       if p_m ≤ P_M then
25           offspring_i ← Mutate(offspring_i)
26       Topology_i ← Translate_Chromosome(offspring_i)
27       Fitness_i ← Network_Simulator(Topology_i)
28       Fitnesses ← (Fitnesses ∪ Fitness_i)
29       Offsprings ← (Offsprings ∪ offspring_i)
30   i ← i + 1;
31 until i ≤ I;
32 P ← Sort_Descending(G_g, Fitness(G_g))
33 O ← Sort_Descending(Offsprings, Fitness(Offsprings))
34 G_{g+1} ← (O_1, O_2, ..., O_{1.R}, p_1, p_2, ..., p_{1.(1−R)})
35 Best_Topology ← O_1;
36 Best_Fitness ← Fitness(O_1)
37 g ← g + 1
38 until g ≤ G;
```

A string representation of the candidate topology is known as genotype. The decoded representation of the string is known as the phenotype. Note that a candidate topology’s genotype that is represented as a chromosome consists of, for example, a bit string — all the variations of the bits in this bit-string of that genotype cover the entire solution set. The search for optimal topology LWSN becomes a complex and intractable problem because of the high number of nodes and their associated parameters. Finding the optimal candidate would need a large number of computations. Therefore, GA based heuristics solution can be valuable in mitigating this problem.

As discussed above, for OpToGen, a chromosome represents an LWSN topology. The chromosome has been designed to represent sensor nodes (tier-2) as indices of the bit-codes. The bit values at each index represent either the sink node or relay node (tier-1). The architect is assumed to know the physical length of the LWSN beforehand, for instance, if the architect is planning to cover a highway or certain length with an LWSN of the camera and other sensor nodes. The architect would also know the field of view of the camera and detection ranges of the sensor nodes. Therefore, the architect would know the number of cameras and nodes required to cover the highway. The number of nodes, N, is one of the inputs of OpToGen. If we assume that the architect places one relay node between every two pairs of sensor nodes for maximum...
reliability, the number of relay node can be calculated to be
the maximum of half of \( N \). These values, along with the user-
selected probability of crossover, \( P_c \), and the probability of
mutation, \( P_m \), are given as input to OpToGen. Hence, we can
list the assumptions as:

1) Sensor nodes can be less than or equal to twice the
number of relay nodes.
2) Data can be sent directly to the sink node by a sensor
node if it is in the transmission range.
3) Sensor nodes forward their data through a relay node
if the sensor node is not in range of the sink node.
4) All relay nodes are powerful enough to be directly
connected to the sink.

Let us take the example of an LWSN with \( N = 6 \). Assuming
that the number of relay nodes is a maximum half of \( N \), so
we get three relay nodes and one sink. Therefore, our sample
chromosome represents a network with one sink, three relay
nodes, and six sensor nodes. Bit-string that represents this
network has a length that can be calculated using Eq. 1.

\[
N \times \left( \left\lceil \log_2 \left( 1 + \frac{N}{2} \right) \right\rceil \right) \quad (1)
\]

From our example of a network of \( N = 6 \), Eq. 1 calculates
bit-string length as 12-bits, where the indices of the bit pairs
will represent the sensor node number and the value of the bits
will represent the connection. This means that bit-1 and bit-2
represent sensor node 1’s connectivity; bit-3 and bit-4 represent
sensor node 2’s connectivity and so on. The connectivity is
determined by Eq. 2.

\[
connection \equiv \text{base}_{10} \text{(bitcode)} \mod \left( 1 + \left\lceil \frac{N}{2} \right\rceil \right) \quad (2)
\]

Let us elaborate on the use of Eq. 2 with the help of Table
I. Assuming the bit string has the value of 01111101000, we
break it down to sets of two bits each.

The first row of Table I contains indices of individual bits
of the representing chromosome. The values of the bits are
in the second row. The third row contains the ID of a sensor
node whose connectivity is being depicted by the bits. Finally,
the decoded connection, according to Eq. 2, is shown in the
fourth row.

OpToGen designates the connection value of 0 to sink, the
connection value of 1 to Relay 1, the connection value of 2 to
Relay 2 and so on. Therefore, using Eq. 2, OpToGen decoded
bit code 00 as sink i.e. \( \text{base}_{10}(00) \mod 2 \equiv 0 \). Bit code
01 is decoded as Relay 1, i.e. \( \text{base}_{10}(01) \mod 3 \equiv 1 \). Similarly, bit code 10 is decoded as Relay 2 and bit code
11 as Relay 3. Therefore, the network represented by the
chromosome sample in Table I has Sensor Node 1 and Sensor
Node 2 connected to Relay 1. Sensor Node 3 connected to
Relay 3. Sensor Node 4 and Sensor Node 5 are connected to
Relay 2. Lastly, Sensor Node 6 is directly connected to the
sink. This network is logically depicted in Fig. 6. Note that the
actual physical network may be different. Also, note that all
relay nodes are assumed to be connected to the sink; therefore,
their representation in the chromosome is not required.

Let us take the example of another network, where \( N = 9 \).
Using Eq. 1, the bit-string length is calculated to be 27. Table
II depicts a sample chromosome of length 27. Assuming
the bit string has the value of 01000100111010110100100,
we break it down to sets of three bits each.

Using Eq. 2, bit code 000 corresponds to sink. Bit code 001
 corresponds to Relay 1. Bit code 010 corresponds to Relay 2.
Bit code 011 corresponds to Relay 3. Bit code 100 corresponds
to Relay 4. Bit code 101 corresponds to Relay 5. Bit code 110
 corresponds to sink again, because \( \text{base}_{10}(111) \mod 6 \equiv 0 \). Bit code 111 corresponds to relay-1 because \( \text{base}_{10}(111) \mod 6 \equiv 1 \).

Therefore according to the decoded chromosome, Sensor
Node 1 and Sensor Node 7 are connected to Relay 2. Sensor
Node 2 and Sensor Node 4 are connected to Relay 1. Note
that although their bit codes are different, yet both bit code
001 and bit code 111 when given as input to Eq. 2 give Relay
1 as output. Sensor Node 3 is directly connected to the sink.
Sensor Node 5 and Sensor Node 6 are connected to Relay 3.
Sensor Node 8 and Sensor Node 9 are connected to Relay 4.
There is no sensor node connected to Relay 5; therefore, it
does not take part in data forwarding. The logical network is
represented in Fig. 7. Note that both chromosomes of Table I
and Table II are sample chromosomes chosen randomly. They
do not represent the optimal solution.

The function \textit{Translate Chromosome()}, as shown in Algo-


rithm 1, is responsible for chromosome translation. It takes
the bit-string as input and converts it into parameters of network
topology that understandable by the network simulator. In
the next step, function \textit{Network Simulator()} calls the network
simulator and gives the topology and other network parameters
as input to it. The network simulator evaluates the fitness of
input topology by running a network simulation and extract-


ing results. The output of \textit{Network Simulator()} consists of a
measure of energy remaining in the simulated network. This is
called Total Residual Energy (TRE). Using TRE as input, the
fitness of a network represented by a chromosome \( x \) is rated
according to Eq. 3.

\[
\text{Fitness}(x) = \min(-TRE(x)) \quad (3)
\]

As \( TRE \) is the total remaining energy of the topology simu-
lated by \textit{Network Simulator()}, a lower value of \( TRE \) represents
lower network lifetime. A high value of \( TRE \) would mean that
the nodes of the network have more energy remaining after the simulation has run. Therefore, according to Eq. 3, higher TRE represents a lower value of fitness, which results in better fitness.

C. Genetic Operations

This subsection explains how genetic operations are carried out in OpToGen. EC requires evolutionary or genetic operators to simulate the process of evolution. Parents are selected from the current population (i.e., set of candidate solutions) for breeding. We use Select_Random_Parent() to select a parent for breeding in the current implementation of OpToGen. This function can be refined in future versions of OpToGen. After two parents are selected, firstly, a crossover between the parents is carried out based on the probability of crossover, \( P_C \). The resultant offspring are subjected to mutation based on the probability of mutation, \( P_M \). Finally, the current population is extended by adding this mutated offspring. These genetic operations are applied \( I \) times.

The genetic operations of crossover and mutation result in an updated population of size \( 2I \). These are candidates for the next generation. Now, the fitness of newly added offsprings needs to be assessed. This is done by translating the offspring chromosomes into network topologies using Translate_Chromosome() and then evaluated using Network_Simulator(). The parents of the current generation were assessed at the start of the algorithm. Both parents and offsprings are sorted based on fitness using Sort_Decending(). The next generation will consist of \( R\% \) of stronger offsprings and \((100 - R)\% \) of stronger parents. This results in a new generation of size \( I \).

Sorting also gives us the best (Best_Topology) along with its fitness (Best_Fitness) for each generation. The above process is carried out \( G \) times. Best_Topology either remains same or improves with each iteration.

IV. Evaluation

To quantify the usefulness for OpToGen, we implemented OpToGen in Matlab. We used NS-2 network simulator. The OpToGen’s implementation in Matlab takes the network parameters as input and generates the initial population. Then it saves the network parameters in a text file that is used with TCL script to run NS-2. The implementation is OS independent. The results of NS-2 are then written into a text file that Matlab uses for further processing.

A. Experiments with different Combinations of \( P_C \) and \( P_M \)

Matlab was used for OpToGen’s performance evaluation. Best fitness corresponds to optimum network topology. For this purpose, we performed experiments with the parameter shown in Table III.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range / Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N )</td>
<td>6</td>
</tr>
<tr>
<td>( G )</td>
<td>30</td>
</tr>
<tr>
<td>( I )</td>
<td>20</td>
</tr>
<tr>
<td>( P_C )</td>
<td>0.6 - 0.9</td>
</tr>
<tr>
<td>( P_M )</td>
<td>0.1 - 0.5</td>
</tr>
<tr>
<td>( R )</td>
<td>30%</td>
</tr>
</tbody>
</table>

We ran twenty simulations for each pair of \( < P_C, P_M > \) to find average fitness per \( < P_C, P_M > \) pair, best fitness per \( < P_C, P_M > \) pair and best fitness per generation per \( < P_C, P_M > \) pair.

Fig. 8 - Fig. 10 show the results of experiments to find best fitnesses mentioned above against different values of \( P_C \) and \( P_M \). In these figures, \( P_C \) is along the x-axis, \( P_M \) is along y-axis, and best fitness is along the z-axis. The surface contains points for all experiments.

According to Fig. 8, the first set of 20 experiments is performed with \( P_C = 0.6, P_M = 0.1 \) and the topology corresponding to best fitness of -4.66 was found. Note that, this is the average of best fitnesses of all 20 experiments with the same \( P_C, P_M \) pair. Similarly, the second set of 20 experiments is performed with \( P_C = 0.6, P_M = 0.2 \), and the best fitness is found to be -4.67 and so on.
Fig. 9 indicates best fitnesses per generation for all these experiments, and Fig. 10 shows mean fitness for the same. In this way, we show the convergence of OpToGen using different $<P_c, P_m>$ pairs.

During these experiments, it was seen that with the pair $<0.6, 0.3>$, a chromosome with the best fitness of 4.7 was created by OpToGen in the 9th generation. This means that in only $9 \times 20 = 180$ runs of NS-2, OpToGen converged to the best fitness.

Here, we would like to compare the resultant topology of OpToGen with a random topology from the first generation (depicted in Table I). The bit string with the best fitness had the value of 100101100000. Table IV shows the decoded value of the chromosome, and Fig. 11 logically depicts the network with the best fitness. This can be compared to the random topology depicted in Fig. 4 and explained through Table I. The fitness of the random topology with chromosome 01111101000, when running through NS-2, gave a value of -3.1.

B. Comparison with Brute Force

Now, we look into the probability of finding the best fitness. For this, we compared the best fitness achieved from OpToGen keeping $N = 6$ for all possible combinations of networks. As chromosome’s bit-string length is 12, brute force search was carried out by running NS-2 up to $2^{12}$ (i.e., 4096) times, covering all possible topology variations.

As depicted in Fig. 12, it took 3062 out of 4096 simulations to find out the chromosome with the best topology.
TABLE IV. CHROMOSOME REPRESENTING TOPOLOGY WITH BEST FITNESS

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>1-2</th>
<th>3-4</th>
<th>5-6</th>
<th>7-8</th>
<th>9-10</th>
<th>11-12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome</td>
<td>10</td>
<td>01</td>
<td>01</td>
<td>10</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>Sensor Node</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Decoding</td>
<td>Relay-2</td>
<td>Relay-1</td>
<td>Relay-1</td>
<td>Relay-2</td>
<td>Sink</td>
<td>Sink</td>
</tr>
</tbody>
</table>

Fig. 13. Histogram of Fitnesses

A histogram of fitnesses of all 4096 simulations is shown in Fig. 13. Out of 4096 chromosomes, 3312 chromosomes had a fitness between -3 and -4, whereas, 23 chromosomes lie below -4.5. Table V shows the probability of choosing a random chromosome from all possible combinations.

TABLE V. PROBABILITY OF RANDOMLY SELECTING A CHROMOSOME WITH OPTIMAL FITNESS

<table>
<thead>
<tr>
<th>Range</th>
<th>Count</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3 to -4</td>
<td>3312</td>
<td>80 %</td>
</tr>
<tr>
<td>-4.5 to -4.75</td>
<td>20</td>
<td>0.49 %</td>
</tr>
<tr>
<td>less than -4.75</td>
<td>3</td>
<td>0.00073 %</td>
</tr>
</tbody>
</table>

Hence, there was a probability of 0.00073% that a randomly selected chromosome had optimal topology. This shows how quickly OpToGen converged the best or near best as compared to brute force search.

C. Evaluation of Residual Energy

Let us compare the lifetime of OpToGen’s best topology with the topologies of networks with mean fitness and least fitness, respectively. Fig. 14 shows the residual energy of three networks. The red dashed-and-dots line shows how and when the energy of the topology with the least fitness was consumed. It can be seen that the network was entirely depleted by 22-time units of NS-2 simulation. The blue dashed line shows the residual energy of the network with mean fitness. The mean fitness topology had a lifetime of 26.5-time units of NS-2 simulations. On the other hand, the best topology that OpToGen chose (depicted in Table IV earlier) lasted till 28-time units. In these cases, the topology from OpToGen ran 27% of the time more than the least fitness topology.

D. Fitness Convergence

Table VI shows the number of NS-2 runs taken by OpToGen to achieve best fitness for all \( P_C, P_M \) pairs. For \( P_C = 0.6 \) and \( P_M = 0.1 \) best fitness is achieved in the 18th generation. OpToGen performed 360 NS-2 runs to achieve best fitness and so on. Table VI depicts suitable \( P_C, P_M \) for OpToGen to find best fitness.

E. Energy Consumption

Energy consumption is an important parameter for performance evaluation LWSNs. We have calculated residual energy...
of the LWSN, total energy consumed by the LWSN, and total energy consumed per node of LWSN for all above experiments.

1) Energy Consumption based on $P_C$ vs $P_M$: Fig. 15-17 show energy consumption for experiments discussed above. For the first set of experiments with $P_C = 0.6$ and $P_M = 0.1$, residual energy for the LWSN was 3.73 units depicted in Fig. 15. Total energy of the LWSN was found to be 96.26 units corresponding to best fitness depicted in Fig. 16. Total energy per node was found to be 9.6 units as depicted in Fig. 17.

2) Energy Consumption vs Network Size: A set of experiments was performed to analyze the behavior of energy consumption for different network sizes. Very few efforts have been made in literature in this regard, especially for linear wireless sensor network. We repeated the above experiments for linear networks of 6 nodes to 12 nodes to calculate the total amount of energy consumed in these networks. Fig. 18 shows the findings of these experiments. It can be noticed that as the amount of energy consumed is high for large networks and the trend is the same throughout.

3) Number of Generations vs Network Sizes: The final set of experiments was carried out to evaluate OpToGen’s efficiency. We extracted the number of generations OpToGen took to achieve the best fitness for various network sizes. Fig. 19 shows the results of these experiments. The overall trend of the number of generations required to achieve the best fitness decreases for large networks. Therefore to achieve optimal topology in smaller networks takes the number of generations than large networks. One of the reasons for this trend is the linear nature of large networks due to which optimal topology cannot undergo much modification.

V. CONCLUSION

WSNs are being used to monitor structural deployments in smart cities, especially as part of intelligent transportation systems. WSNs for bridges, highways, periphery monitoring, etc. consist of a special class of WSNs called Linear Wireless Sensor Networks. Typical WSN research has focused on optimization issues related to mesh and ad hoc networks. This has left some room in the research of finding optimal topologies for LWSNs.

This paper discussed the details of OpToGen framework that can be used to find optimal topologies of LWSNs using GA. We implemented OpToGen using Matlab and NS-2. The paper also demonstrates that OpToGen proposes optimal topology in much fewer iterations than the time spent to
search the best topology, exhaustively. There are three main contributions of this paper: First, we performed experiments to make the selection of optimal probabilities easy to achieve the best fitness in minimum simulator runs. Second, the trade-off between energy consumption and different network sizes is evaluated for linear wireless sensor network. Third, we have evaluated the efficiency of OpToGen by extracting the number of generations it took to achieve the best fitness for various sized LWSNs.

VI. FUTURE WORK

There are many areas of OpToGen that require further research. These include better parent selection mechanisms and different evolutionary operators. Another area of further research is to find better ways to generate the initial population. In our current work, OpToGen’s fitness function was based on network lifetime. In the future, other network parameters such as throughput, network delays, bandwidth, etc. can be made part of a multi-objective fitness function.

ACKNOWLEDGMENTS

This work is funded by the Grant Number AT-34-335, King Abdul Aziz City of Science and Technology (KACST) of the Kingdom of Saudi Arabia. The authors thank KACST for their support.

REFERENCES

[1] Zhengjie Wang, Xiaoguang Zhao and Xu Qian; The application and issue of linear wireless sensor networks, International Conference on System science, Engineering design and Manufacturing informatization, 2011, Guiyang, pp. 9–12


