Real Time Broadcast Scheduling in Tsch for Smart Healthcare Using Cuckoo Search Algorithm

Pavithra kotteswaran (pavithra.2016@vitstudent.ac.in)  
VIT University - Chennai Campus
rekha D  
VIT University - Chennai Campus

Research Article

Keywords: Internet of Things (IoT), Smart Healthcare, scheduling, Cuckoo Search Algorithm (CSA), Time Slotted Channel Hopping (T SCH), Medium access control

Posted Date: January 19th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-1453501/v1

License: © This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License
Abstract

Smart Healthcare Sensor Network (SHSN) has caught the attention of scientists and researchers. SHSN has a low bandwidth which causes low successful packet transmission. To transmit and receive the data without any collision and for efficient broadcast scheduling Time Slotted Channel Hopping (TSCH) can be used. The broadcast scheduling problem needs to be addressed in SHSN as the bandwidth is limited. This is done by collaterally transmitting the node data in the same time slot such that it does not cause interference with each other. This further minimizes the network turnaround time by enhancing the time slot in the TSCH frame. The proposed work focuses to achieve maximum throughput, turnaround time of nodes by using Genetic Algorithm (GA), Immune Genetic Algorithm (IGA), and Cuckoo Search Algorithm (CSA). The point is to limit the TSCH cycle period and augment the hub communication with diminished calculation time. In examination to GA and IGA, CSA effectively points to improving the arrangements and is unequivocally worried about abusing all accessible information about the issue. The reproduction results on various issue examples affirm that CSA altogether beats a few heuristic and transformative calculations by tackling notable benchmark issues in terms of arrangement quality, which likewise exhibits the viability of CSA in proficient utilization of channel transfer speed.

Introduction:

In today's world, with an increasingly aging population and a large number of chronic diseases, there is a need for smart healthcare systems. With the evolution of the Internet of Things (IoT) in medical institutes are now enabled for providing quality, impromptu and convenient health care services. Healthcare institutes are faced with issues like patient prioritization immediate action in minimum response time [1–4]. The fast growth in the electronic field is mainly in devices like IoT and smartphone communication so on. IoT has become a need for day-to-day life. In smart healthcare, IoT networks can be connected and made to communicate to transfer health data related to the disease. Sensors that are connected in the network sense the values based on the source and capture them. The human body undergoes frequent changes like body temperature, humidity, airflow sensor, electrocardiogram sensor, electromyography sensor; so on... such change can affect much life of people. The manual approach used by healthcare institutes utilizes more time. This causes untimely delayed delivery of information. To manage such a situation SHSN uses an efficient communication medium to share the data. The SHSN can be a single or multi-hop network. The single-hop sensor network nodes communicate directly with each other whereas multi-hop nodes in the connectivity collect and forward a data packet to its intermediate nodes. The smart healthcare sensor signals have limited bandwidth and propagation delay due to the huge network size [5–7]. TSCH is a wireless communication technique that combines time-slotted excess channel hopping with multi-channel abilities. TSCH is considered a fusion method of TDMA and FDMA. TSCH distinctively distribute scheduled slot for transmission to produce maximum throughput in a huge traffic network. It's known to work well in a long duration of time effectively which in turn is avoiding collisions. In TSCH, the nodes are involved in the network to transmit in their timeslot at different frequencies. TSCH based MAC protocols are used to prevent time delay for subsequent node transmission. For optimization of broadcast node scheduling problems, an Evolutionary algorithm is used for application [8–10].

Using three Evolutionary algorithms as a scheduler an evaluation of the proposed CSA and an assessment of the proposed CSAs presentation is then made, followed by a series of recreations to confirm the preferred CSA’s dominance over CSA, GA, and IGA. This work focuses on reducing the turnaround wait time for energy transmission and optimum utilization of the bandwidth of SHSN. The paper is arranged as follows broadcast node scheduling and conflicts in section 2. Section 3 allows explores the problem formation of smart healthcare broadcast node
scheduling. In section 4 CSA, GA, and IGA processes for solving scheduling problems are investigated. Section 5 covers the simulation results of the smart healthcare scheduling and section 6 serve as the conclusion.

**Smart Healthcare Node Scheduling Conflicts:**

The leading packet collision problem in broadcasting in a multi-hop static SHSN is due to two major reasons. Firstly, in a network when the adjacent nodes transmit simultaneously it will cause a primary collision. Secondly, if more packets are transmitted to a node at the same time slot it causes a secondary collision. The secondary collision can happen when transmission occurs at the same time in two-hop away nodes from the node that was sent. Both the primary and secondary conflicts faced the exposed and hidden terminal problems. The solution to this is allowing the two nodes to transmit without any conflict in the same time slot, such that the nodes are located in two-hops away. Another major issue in smart healthcare scheduling problems is propagation delay. There is a continuous change in healthcare sensor data reading of patients like such as temperature, ECG, EMG, and so on. This propagation delay can be bypassed by computing a maximum size of the accepted value, guard time, and a packet with extreme transmission length [11].

Smart healthcare sensor communication uses TSCH-based MAC protocol, which adds a guard time after every packet sent to avoid packet collisions. The guard time avoids a delayed reception or early reception of a packet in the same propagation delay can be minimized by staggering transmission with the use of propagation estimation. Smart healthcare scheduling is constrained by half-duplex communication where a node can either send or receive, but not perform both functions at the same slot of time. The main focus of the work is to create a smart healthcare TSCH broadcast scheduling which has an average low turnaround time for nodes that are subsequently transmitting within a minimum utilization of smart healthcare sensor node bandwidth by scheduling the transmission in the TSCH frame of all the possible collateral nodes [12–14].

**Creation Of Smart Healthcare Broadcast Node Scheduling:**

SHSN is depicted by an undirected graph network $G (N, V)$ where $N$ stands for smart healthcare sensor nodes present in the network and the $V$ stands for a link between every node that is bidirectional. The nodes $k$ and $l$ are linked within transmission range which is conveyed via link $E(k, l)$ Fig. 1 shows a simple multi-hop sensor network where every node is transmitting to the subsequent node to form a link among them [16].

$N = a,b,c,d,e,f,g,h, i$ is a representation of nodes in the given network, and $N$ is the number of nodes in the network. The primary and secondary collision has to be avoided to dismiss broadcast conflicts. By identification of the connectivity matrix in the network, a primary collision can be avoided. Further, secondary collisions can be avoided by identifying two-hop connectivity in the network, thus avoiding collisions in the same time slot and allowing parallel node transmission to be increased Fig. 2 illustrates the connectivity matrix (CM) computed from Fig. 1.

In CM, each row depicts direct communication in nodes and each column depicts the smart healthcare sensor node in the network. The matrix has 0 and 1 where 1 represents the existing communication and 0 represents the no communication of the network. The Hop Matrix (HM) shown in Fig. 3 represent the two-hop connectivity in the smart healthcare sensor network. This matrix has a value of one or two in the row which represents a hop connection that occurs between every node and the column represents the number of smart healthcare sensor nodes in the network. The row contains 0 or 1 where 0 means no communication and 1 means one or two hop communication [17–19].
In the Scheduler Matrix (SM), \( M \) = the number of time slots, \( N \) = number of smart healthcare sensor nodes \((n_1, n_2, n_3, \ldots, n_n)\) is the number of nodes in the network. This is shown in Fig. 4 and Fig. 5. In the matrix, the number of a timeslot is shown in the row and the joining node in the network are shown in the column. Rows and columns take the values 0 (or) 1 where 1 shows the transmitting nodes in a timeslot without any collision. The TSCH frame for the SHSN is shown in Fig. 4 and Fig. 5 shows the optimum GA-TSCH framework that is possible for collateral node transmission with a minimum timeslot for the network [20–21].

Smart healthcare scheduling suffers from non-deterministic polynomial (NP) completely. This problem cannot be solved efficiently. For an optimal solution to the NP problem, GA is used. Smart healthcare scheduling has broadcast more scheduling problems for which tight lower bound is used to check that CSA has got the optimum results.

Tight lower bound is calculated by

\[ ND = \max |Ds(n)| \quad (1) \]

\( n \in N \)

Ds is a set of the degree of nodes \( n \) the highest degree of node transmission in SHSN is represented by ND.

The tight lower bound value is calculated based on the highest degree of node transmission.

\[ \Delta = |M| - ND \geq 1 \quad (2) \]

When the value of \( \Delta \) is 1 optimum solution is reached. \( \Delta \) represents the minimum time frame that is needed for single transmission for every node in the network.

**Evolutionary Algorithm:**

**4.1 Genetic Algorithm in Smart Healthcare**

GA is based on the survival of the fittest Darwinian principle. Evolution and natural selection are the basics of these heuristic search techniques. For NP-complete problems, GAs serve as an effective search technique. This technique uses an evolutionary algorithm as a basis of selection using parameters such as crossover, mutation, inheritance, and selection. GA applies a natural selection mechanism to find optimized results. The optimized solution towards the problem is achieved using computer simulation which involves the use of a group of abstract representations for the candidate solution. A possible set of solutions is generated from individuals from a population. Out of the population, two parents are chosen by the principal of fitness [22–23]. This results in child selection in every generation for the propagation of the next generation. The criteria for stopping the algorithm from further propagation is either reaching the maximum fitness lever or maximum generation.

GA when applied in TSCH matrices uses different permutations for every node to transmit in a different time slot and various frequencies. So TSCH matrix forms the first level of population for this problem. The iteration process is started by selecting two parents from the initial population which is the TSCH matrix. When the algorithm reaches its maximum run time or the tight lower bound is reached \( (\Delta = 1) \), it stops further iterations. The algorithm shows an overview of the GA and every step of iteration performed.

**4.1.1 Selection**
Every iteration is carried out with operators such as selection, crossover, and mutation operators. Half a percentage of chromosomes is selected from the natural population using the natural selection operator for reproduction. TSCH frames have the same calculation using Eq. (3). The various TSCH matrices are collected in a matting pool. Two parents are selected from the matting pool based on their fitness using GA. Good parents are consciously selected so that there is more probability of producing better next generation. HM is used to check for primary and secondary conflicts for each child in every generation this avoids collisions. If the child violates the criteria this child is not allowed into the next generation. Fitness is evaluated for every generation propagated.

### 4.1.2 Crossover

Crossover is a very important optimization technique in GA. The crossover operator chooses random bit strings for sharing information between the individuals after the selection operator selects the parents from the population. Every bits string can exchange within a specific crossover point. Eg. Considering two parents \( p_1 = 1111 \) and \( p_2 = 0000 \) are crossed over at random points the \( c_1 = 0011 \) and \( c_2 = 1100 \) are the offspring of the crossover. The fitness of the offspring is checked in every generation for the optimum solution. For the next generation, the fitter offspring can take the position of the parent. When the fitter individual's selector is joined, the next generation of persons will be formed.

### 4.1.3 Mutation

Premature convergence is prevented in GA through mutation. In mutation, some bits are flipped after crossover. The flipping is done by replacing 0 with 1 and 1 with 0. To avoid conflicts in HM and overcome any deficiency for further generations. The next-generation higher fitness child is selected.

### 4.1.4 Fitness:

Fitness is evaluated for each population based on the following criteria: 1. Smart Healthcare bandwidth utilization (i.e) maximum number of packets successfully transmitted. 2. Tight lower bound value \( \Delta = 1 \). Once the fitness is achieved the GA stops further iterations. These conditions determine that the optimum solution is achieved in average time delay for a smart healthcare network.

### 4.2 Immune Genetic Algorithm:

Crossover and mutation, two important genetic operators in GA, not only give everyone the transforming potential to achieve universal perfection, but they also encourage decadence owing to the irregular and unaided looking throughout the cycle. GA, on the other hand, is the inability to use a key and well-defined trademark or information in a pending issue. Invulnerable Hereditary Calculation was proposed in light of the foregoing considerations. The structure of invulnerable hereditary calculation is shown in Calculation 2.

After the generating step, the setup is made for safe activities. IGA is a clever streamlining computation that, for the most part, creates an invulnerable administrator through two stages: invulnerability assessment and immunization. The data included an IGA calculation that functioned in a certain way [24–25].

### 4.2.1 Immune Selection:
The recently made populace after multiplication, which fulfills the essential and auxiliary requirements, is chosen for copy column disposal. The subsequent populaces are organized by the channel use variable and put away in the immunization pool.

**4.2.2 Vaccinations:**

Immunization is used to improve health by altering the characteristics of an individual population with prior data to achieve higher wellness with a greater likelihood. For inoculation, a chromosome from the immunization pool is taken. The IGA distinguishes which hub sends the populace first. Some other hub, which does not make impedance with the communicative hub, can be allowed to communicate within the same time allocation. To perform this, a hub is chosen arbitrarily and checked with the jump lattice whether it makes an impedance with the as of now communicating hub, if not the hub esteem is transformed to one, permitting the chosen hub to communicate in a similar time-space. The qualities of the chosen chromosome are changed because of the information got from the jump framework of the given system thus the inoculation cycle increments the number of transmissions.

**4.3 Cuckoo Search Algorithm:**

CSA, eggs in the nest are a solution, and cuckoo eggs are a new solution. The quality of the new generation of eggs corresponds to the ability of the eggs to produce new cuckoo. The CSA is constructed based on the following assumptions [26–29].

1. Each cuckoo chooses a host nest and lays eggs randomly.
2. The next generation will choose the best egg nest with the highest quality.
3. The host bird can find cuckoo eggs with the probability \( p = [0, 1] \).
4. If the host bird finds that the egg does not belong to her, the host bird will throw the egg Cuckoo’s nest or abandon the nest and build a new nest in a new location.

**Simulation Results:**

Comparing the performance of CSA with mean-field annealing, genetic algorithm, and Immune genetic algorithm in solving broadcast scheduling problems runs a series of simulations. The simulation results for the number of nodes \( |N| \), the number of slots \( |M| \), and the network degree are discussed in the following sections. Smart healthcare channel utilization for the entire network is defined as a fitness function factor.

\[
\alpha = \frac{1}{M \times N \times F} \left( \sum_{i=0}^{M} \sum_{j=0}^{N} \sum_{k=0}^{F} (TSCHM_{ijk}) \right) \tag{3}
\]

Average time delay

\[
\beta = \frac{N \times F}{M} \sum_{i=0}^{M} \left[ \frac{1}{\sum_{j=0}^{N} \sum_{k=0}^{F} (TSCHM_{ijk})} \right] \tag{4}
\]

For optimal smart healthcare network design, the average time delay for every node is calculated by the average availability of the network in minimum turnaround time.
Using python simulation results arrived out for smart healthcare sensor node on various parameters using Table 1. The number of nodes used various between 10 to 100 nodes. In a ten node simulation setup for smart healthcare sensor nodes, every node 5-time slots are utilized.

### 5.1 Simulation Results For Ga

The motivation behind the first reenactment was to research the execution of hereditary calculation for various organizations appeared in Table 1. The quantity of hubs taken for reenactment goes from five to a hundred. More modest hub organizations performed by more digit of broadcast in a worthy age. Notwithstanding, a 100 nodes organization with 198 edges with the level of nine distinguishes the ideal arrangement TSCH outline after 489 generations. The regular digit of generation for the 100 nodes system is 99.1. This must stay diminished to reduce the performance time.

<table>
<thead>
<tr>
<th>Number of smart healthcare nodes</th>
<th>Number of links</th>
<th>Average degree</th>
<th>Average ND</th>
<th>Maximum ND</th>
<th>Maximum TSCH frame length</th>
<th>Avg (α)</th>
<th>The average number of generations</th>
<th>Computation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>23</td>
<td>4.2</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0.215</td>
<td>31.1</td>
<td>0.582s</td>
</tr>
<tr>
<td>25</td>
<td>45</td>
<td>5.3</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td>0.106</td>
<td>39.4</td>
<td>1.06min</td>
</tr>
<tr>
<td>30</td>
<td>68</td>
<td>5.4</td>
<td>5.3</td>
<td>7</td>
<td>8</td>
<td>0.117</td>
<td>42.5</td>
<td>1.82min</td>
</tr>
<tr>
<td>45</td>
<td>99</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>10</td>
<td>0.123</td>
<td>46.3</td>
<td>2.23min</td>
</tr>
<tr>
<td>60</td>
<td>109</td>
<td>6.4</td>
<td>7.5</td>
<td>9</td>
<td>10</td>
<td>0.112</td>
<td>52</td>
<td>4.59min</td>
</tr>
<tr>
<td>80</td>
<td>155</td>
<td>7.3</td>
<td>8.1</td>
<td>10</td>
<td>11</td>
<td>0.102</td>
<td>64.3</td>
<td>6.25min</td>
</tr>
<tr>
<td>100</td>
<td>198</td>
<td>7.8</td>
<td>8.2</td>
<td>10</td>
<td>11</td>
<td>0.105</td>
<td>99.1</td>
<td>15min</td>
</tr>
</tbody>
</table>
### Table 2
Immune Genetic Algorithm using TSCH simulation results

<table>
<thead>
<tr>
<th>Number of smart healthcare nodes</th>
<th>Number of links</th>
<th>Average degree</th>
<th>Average ND</th>
<th>Maximum ND</th>
<th>Maximum TSCH frame length</th>
<th>Avg (α)</th>
<th>Average number of generations</th>
<th>Computation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>20</td>
<td>3.4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0.259</td>
<td>17.2</td>
<td>0.3s</td>
</tr>
<tr>
<td>15</td>
<td>30</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0.210</td>
<td>18.1</td>
<td>0.49s</td>
</tr>
<tr>
<td>20</td>
<td>35</td>
<td>4.2</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>0.205</td>
<td>18.9</td>
<td>0.69s</td>
</tr>
<tr>
<td>30</td>
<td>60</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>0.193</td>
<td>20.9</td>
<td>18s</td>
</tr>
<tr>
<td>45</td>
<td>90</td>
<td>4.3</td>
<td>6</td>
<td>8</td>
<td>9</td>
<td>0.181</td>
<td>33.2</td>
<td>3.5min</td>
</tr>
<tr>
<td>60</td>
<td>110</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0.171</td>
<td>59.4</td>
<td>10.41min</td>
</tr>
<tr>
<td>75</td>
<td>155</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0.165</td>
<td>65.4</td>
<td>2.19min</td>
</tr>
<tr>
<td>85</td>
<td>170</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0.153</td>
<td>89.1</td>
<td>14.98min</td>
</tr>
<tr>
<td>100</td>
<td>210</td>
<td>5</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>0.116</td>
<td>97.51</td>
<td>25.67min</td>
</tr>
</tbody>
</table>

### Table 3
Cuckoo Search Algorithm using TSCH simulation results

<table>
<thead>
<tr>
<th>Number of smart healthcare nodes</th>
<th>Number of links</th>
<th>Average degree</th>
<th>Average ND</th>
<th>Maximum ND</th>
<th>Maximum TSCH frame length</th>
<th>Avg (α)</th>
<th>Average number of generations</th>
<th>Computation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>90</td>
<td>3.5</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>0.201</td>
<td>4.03</td>
<td>6s</td>
</tr>
<tr>
<td>90</td>
<td>160</td>
<td>3.9</td>
<td>7</td>
<td>10</td>
<td>11</td>
<td>0.163</td>
<td>6</td>
<td>10s</td>
</tr>
<tr>
<td>110</td>
<td>152</td>
<td>3</td>
<td>6.8</td>
<td>10</td>
<td>11</td>
<td>0.121</td>
<td>8.49</td>
<td>1.5min</td>
</tr>
<tr>
<td>120</td>
<td>200</td>
<td>4.2</td>
<td>7.3</td>
<td>11</td>
<td>12</td>
<td>0.130</td>
<td>16.48</td>
<td>1.9min</td>
</tr>
<tr>
<td>150</td>
<td>250</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>12</td>
<td>0.125</td>
<td>17.21</td>
<td>2.5min</td>
</tr>
<tr>
<td>200</td>
<td>410</td>
<td>4</td>
<td>7.5</td>
<td>8</td>
<td>9</td>
<td>0.130</td>
<td>20.33</td>
<td>9.08min</td>
</tr>
<tr>
<td>250</td>
<td>460</td>
<td>5</td>
<td>7</td>
<td>10</td>
<td>11</td>
<td>0.126</td>
<td>28.19</td>
<td>11.2min</td>
</tr>
<tr>
<td>300</td>
<td>600</td>
<td>4</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>0.139</td>
<td>52.14</td>
<td>28.9min</td>
</tr>
<tr>
<td>350</td>
<td>650</td>
<td>5</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>0.111</td>
<td>56.15</td>
<td>32.2min</td>
</tr>
<tr>
<td>400</td>
<td>800</td>
<td>4</td>
<td>8</td>
<td>15</td>
<td>16</td>
<td>0.148</td>
<td>58.12</td>
<td>55.1min</td>
</tr>
<tr>
<td>450</td>
<td>850</td>
<td>5</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>0.121</td>
<td>65.21</td>
<td>62.2min</td>
</tr>
<tr>
<td>500</td>
<td>1010</td>
<td>4</td>
<td>9</td>
<td>14</td>
<td>15</td>
<td>0.129</td>
<td>85.02</td>
<td>65.12min</td>
</tr>
</tbody>
</table>
5.2 Simulation Results For Iga:

For varying numbers of nodes and edges, Table 2 significantly outperforms IGA's output.

When compared to GA, information included in IGA might vastly improve the searching capacity and versatility, as well as dramatically speed up the process. They chose an antigen that is improved with a larger number of transmissions at the same time as the vaccination measure, to use the channel to connect with many nodes at the same time without interference. When comparing the reproduction aftereffects of IGA to GA in Table 1, the quantity of generation is reduced, while the normal number of transmissions of each network is improved. The arrangement is connected to satisfactory age for networks with 85 and 100 nodes. The best arrangement is recognized in 14 minutes and 23 minutes for a 100-node network with normal levels of four and five, respectively. In any event, IGA meets the first two criteria; however, the third, namely, the running time of a large organization, is not reduced.

Table 4 compares the total number of transmissions generated by MA for different node networks with different time slots to GA and IGA.

<table>
<thead>
<tr>
<th>No. of nodes</th>
<th>Time slots [M]</th>
<th>GA</th>
<th>IGA</th>
<th>CSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>7</td>
<td>15</td>
<td>19</td>
<td>33</td>
</tr>
<tr>
<td>25</td>
<td>9</td>
<td>32</td>
<td>35</td>
<td>43</td>
</tr>
<tr>
<td>35</td>
<td>12</td>
<td>47</td>
<td>49</td>
<td>58</td>
</tr>
<tr>
<td>35</td>
<td>13</td>
<td>53</td>
<td>55</td>
<td>62</td>
</tr>
<tr>
<td>40</td>
<td>8</td>
<td>67</td>
<td>69</td>
<td>79</td>
</tr>
<tr>
<td>40</td>
<td>9</td>
<td>81</td>
<td>84</td>
<td>91</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>88</td>
<td>94</td>
<td>103</td>
</tr>
</tbody>
</table>

5.3 Simulation Results For Csa:

CSA aims to achieve convergence speed by simulating the results of aggregation as the number of transmissions increases, while other methods aim at efficiency. The results of IGA and GA can be seen in Tables 1 and 2, which demonstrate a reduction in generation size and computation time while increasing channel utilization. Table 3 shows the average channel use, average number of generations, and calculation time for various networks that use CSAs at various degrees of networks. CSA takes 1.5 minutes for a 110-node network, which is faster than IGA, which takes 26.61 minutes for the same network. IGA and other recently proposed efficient methods do not have much efficiency for networks with more than 100 nodes compared to CSA. This is the key benefit of CSA.

By comparing CSA and GA, it is faster to compute optimal solutions and generate more generations to identify optimal solutions. In Fig. 6, CSA and modified genetic algorithm are compared by the average time taken. Figure 7
compares the time delay of different calculations with CSA.

CSA Algorithm

**Objective function:**

Generate an initial population of $n$ host nests;  
While ($t < \text{Max Generation}$) or (stop criterion)  
Get a cuckoo randomly (say, $i$) and replace its solution by performing Lévy flights;  
Evaluate its quality/rate  
if ($F > F_j$),  
Replace $j$ by the new solution;  
end if  
A fraction ($p_a$) of the worse nests are abandoned and new ones are built;  
Keep the best solutions/nests;  
Rank the solutions/nests and find the current best;  
Pass the current best solutions to the next generation;  
end while

**Experimental Result:**

An image of IoT nodes is shown in Fig. 8. This proposed project was implemented in an IoT environment with 21 nodes. As shown in Fig. 9, the nodes are randomly placed, and their one-hop and two-hop connections are shown. A CM, HM, and SM network structure is calculated based on the network structure. Taking the SM as input, the proposed algorithm in section 5 is employed to increase the number of transmissions. Figure 4 shows optimization results for the implemented real-time scenario in terms of average degree, throughput, and computation time. As can be seen in the table, the 21-node network provides the maximum throughput.

<table>
<thead>
<tr>
<th>No. of nodes</th>
<th>Average degree</th>
<th>Throughput</th>
<th>Computation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>3.2</td>
<td>30</td>
<td>1.01s</td>
</tr>
<tr>
<td>21</td>
<td>3.3</td>
<td>32</td>
<td>0.9s</td>
</tr>
<tr>
<td>21</td>
<td>3.3</td>
<td>30</td>
<td>1.2s</td>
</tr>
<tr>
<td>21</td>
<td>3.2</td>
<td>32</td>
<td>1.3s</td>
</tr>
<tr>
<td>21</td>
<td>3.9</td>
<td>29</td>
<td>1.4s</td>
</tr>
</tbody>
</table>

**Conclusion:**

Discussion and solution of IoT multi-hop network broadcasting schedule taking into account genetic algorithm, IGA, and CSA in specific fields. When compared with GA, IGA and CSA are actively working on improving the solution, while GA is barely watching the search space. In the CSA, all available knowledge of the problem is utilized, whereas, in the immune genetic algorithm, knowledge is obtained from the hop matrix during vaccination. CSA overcomes the
problem by increasing transmissions in a reduced time slot but not in a time slot with adequate calculation time. The main disadvantage of the previous paper was the computation time for large networks, which is greatly reduced by this paper. In a simulation, the effectiveness of CSA for broadcast scheduling problems was verified in terms of channel utilization, algebra, and run time. CSA reaches a strict lower limit in a shorter run time compared to other algorithms. The result verified CSA’s efficiency and effectiveness.

**Declarations:**

**Research Involving Human Participants and/or Animal:**

This article does not contain any studies with human participants or animals performed by any of the authors.

**Author contributions:**

All author contributed to this article

**Funding:**

No funding resource

**Data availability:**

Not used any dataset for this article

**Conflict of interest:**

The authors declare that they have no conflict of interest.

**References:**


Figures

Figure 1

Multi-hop network

\[
\begin{pmatrix}
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\end{pmatrix}
\]

Figure 2

Connectivity Matrix
Figure 3
Hop Matrix

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

HM=

Figure 4
TSCH Matrix

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

TSCHM=

Figure 5

optimum TSCH frame
Figure 6
comparison of computation time by GA and CSA

Figure 7
comparison of average time delay by GA IGA and CSA
Figure 8

Real Nodes

Figure 9

Taken network from experimental result