IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 November 2022, accepted 5 December 2022, date of publication 12 December 2022,
date of current version 16 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3228585

RESEARCH ARTICLE

Game of Sensors: An Energy-Efficient Method to Enhance Network Lifetime in Wireless Sensor Networks Using the Game of Life Cellular Automaton

JOEL REYES[1], FRANCISCO GARCÍA[1], MARÍA ELENA LÁRRAGA[2], JAVIER GÓMEZ[1], AND LUIS OROZCO-BARBOSA[3], (Member, IEEE)

[1]Department of Telecommunications, National Autonomous University of Mexico, Mexico City 04510, Mexico
[2]Institute of Engineering, National Autonomous University of Mexico, Mexico City 04510, Mexico
[3]Albacete Research Institute of Informatics, University of Castilla-La Mancha, 02071 Albacete, Spain

Corresponding author: Francisco García (fgarcia@fi-b.unam.mx)

This work was supported in part by the Research Funds from the Dirección General de Asuntos del Personal Académico (DGAPA)-Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) under Grant IA102822, Grant IA105520, Grant IN119820, and Grant IN101922; in part by the Consejo Nacional de Ciencia y Tecnología (CONACyT) under Grant 320403; in part by the Spanish Ministry of Science, Education and Universities; in part by the European Regional Development Fund; and in part by the State Research Agency under Grant PID2021-123627OB-C52.

ABSTRACT Wireless sensor networks (WSNs) have increased in popularity since they play a relevant role in many applications, such as environmental monitoring, fire detection, and health care, to mention a few. These applications periodically collect data that is relayed to a single sink employing a many-to-one communication pattern. This pattern requires energy-efficient routing protocols since sensors closer to the sink node deplete their energy faster than those sensors located further away. As a result, several techniques have been proposed to solve this problem. For instance, some pieces of research split the network into concentric coronas to provide more energy resources in areas with heavier traffic. However, these techniques use either a predefined network deployment that is not well-suited for all sensor applications or do not always guarantee homogeneous energy consumption. This paper proposes a simple energy-aware routing method based on the Game of Life cellular automaton, which provides a homogeneous energy depletion while extending the network lifespan by considering factors such as residual energy, number of active neighbors, and a sleep schedule. To this end, a discrete dynamic model that takes into account different behaviors of WSN through a set of rules combined with a variation of the A-star algorithm is proposed. Simulation results show that the proposed model accurately balances the energy consumption rate while expanding the network lifetime compared to most recent research works. Furthermore, the proposed method can be combined with path-planning algorithms to improve energy consumption in sparse WSNs.

INDEX TERMS Energy-efficiency, cellular automata, game of life, energy hole problem, path planning.

I. INTRODUCTION

Wireless sensor networks (WSNs) have emerged as a crucial technology capable of carrying out complex tasks such as environmental monitoring, animal tracking, and smart city designing, among many other tasks. Sensor nodes in WSNs can collect data from the physical world and report it to a sink node via single-hop or multi-hop communication. In a multi-hop pattern, sensor nodes around the sink node carry heavier traffic loads than nodes located in the outer areas. This situation implies that sensors closer to the sink node deplete their energy faster, leading to the likelihood that data may not be delivered to the sink node. Thus, implementing energy-efficient routing protocols is a crucial concern in WSNs since energy resources in sensor nodes are supplied

The associate editor coordinating the review of this manuscript and approving it for publication was Kai Yang.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

by small batteries that cannot be easily replaced or recharged, especially in hostile environments.

Nowadays, several techniques have been proposed to deal with the energy consumption problem in WSNs. These techniques can be classified into four categories: radio optimization, sleep/wake-up schemes, energy harvesting, and energy-efficient routing. For instance, radio optimization schemes vary the radio module's parameters such as transmission power [1], modulation, and coding to avoid battery depletion. Sleep/wake-up schemes, on the other hand, switch between active and inactive periods to optimize power consumption depending on the delay requirements, traffic load, and topology characteristics. Energy harvesting schemes use recent advances in microelectronics to develop new sensors capable of collecting energy from their environment through wind, solar radiation, or kinetic energy. Finally, it is well known that transmitting data is ranked as one of the most energy-consuming tasks in WSNs [2]; thus, a crucial concern in WSNs is creating energy-efficient routing schemes that can enhance network lifetime. Several mechanisms have been proposed for this end, and most of them base their operation on clustering. Each cluster usually selects a leader, often called cluster head (CH), responsible for collecting local data and sending it to a sink node directly or through other CHs. Although these proposals have addressed the energy consumption problem, they have usually applied complicated techniques to select CHs that do not necessarily guarantee energy balance among sensors. Other schemes split the network into concentric coronas to provide more energy resources in areas with heavier traffic. This situation balances the energy consumption so that all sensor nodes die at the same time without leaving residual energy [3], [4]. However, these schemes require a predefined network deployment that does not fit all sensor scenarios well. In addition, most of these techniques balance energy consumption by only considering a constant data rate [4], [5]. Other mechanisms use fuzzy logic to solve the uneven energy consumption problem by combining routing protocols with fuzzy techniques. However, these proposals do not always guarantee an even energy consumption.

Consequently, researching and developing algorithms and methods based on different approaches to extending the network lifetime in WSN remains a topic of great interest. In this context, multiple techniques based on cellular automata (CA) [6], [7], [8], [9], [10], [11] have recently been proposed to reduce energy consumption in WSNs. However, most of them base their operation on clustering that uses complicated techniques to select CHs. In contrast, this paper proposes a simple energy-aware routing method based on a cellular automaton that provides a homogeneous energy depletion while extending the network lifespan without clustering.

A cellular automaton (CA) is a decentralized modeling paradigm consisting of entities known as cells. Each cell has a state, a neighborhood, and a finite number of transition rules. By iterating the transition rules over time, a cellular automaton can simulate sophisticated phenomena such as urban traffic and water flow, for example. A CA can also be used to model a WSN, in which a sensor (a cell) can take either an active or inactive state while its neighborhood is composed of sensors within its communication range. In this paper, the active state represents a period in which a sensor can sense the medium and transmit/receive packets. In contrast, the inactive state represents a period in which a sensor sleeps as much as possible.

In particular, our proposed model is based on one of the most interesting CAs known as the Game of Life (hereafter referred to as GoL) [12], [13] whose transition rules are as follows:

- A live cell with fewer than two live neighbors dies from isolation.
- A live cell survives if it has two or three live neighbors.
- A live cell dies from overcrowding if it has more than three live neighbors.
- A dead cell with exactly three live neighbors becomes alive.

In this context, as long as the Game of life CA evolves, some cells will remain in their existing state while others will change state. When modeling WSNs, this evolution of the GoL directly affects energy consumption and network lifespan. Thus, the aforementioned rules can reduce energy consumption in WSNs and avoid redundant information being sent to the sink. However, as time passes, complex patterns formed in the Game of Life can result in holes (i.e., a large set of sensors in an inactive state). This situation can cause failures in routing schemes and data loss. To prevent this problem, this paper proposes new rules based on the Game of Life cellular automaton that considers factors such as the remaining energy, the active number of neighbors, and a sleep schedule to extend network lifetime. In contrast to the existing research, which uses fuzzy logic or corona-based networks, the proposed method presents a straightforward cellular automaton model that intrinsically allows turning sensor nodes off/on. In this way, the network lifespan is improved while balancing energy consumption.

Since GoL is a bio-inspired model that cannot relay packets, the A-star algorithm is adapted to the proposed model with the aim to minimize the number of hops to relay data to the sink node in a dynamic manner. As a result, a dynamic routing system is established that avoids obstacles and makes it possible to reach the destination node in a many-to-one network. The fusion between the new set of rules of the GoL CA and the new variation of the A-star algorithm is called the Game of Sensors (GoS).

Simulation experiments show that the GoS method improves energy balance while enhancing network lifetime as well as the number of transmitted packages through the network compared to the most recent proposals. Moreover, the model preserves the simplicity of cellular automatons models which makes it adequate for large-scale networks.

The rest of this paper is organized as follows: Section II provides an overview of relevant works related to

energy-aware routing schemes. Section III presents the cellular automaton model, as well as the Game of Life model, while Section IV delivers the proposed method (GoS). Section V describes the conducted simulation experiments to validate the proposed method, as well as a comparison with the most recent related works. Finally, Section VI presents the conclusions.

II. RELATED WORK

The problem of maximizing the network lifespan in WSNs has recently received significant attention. The most studied scenario dealing with this issue is the energy-efficient routing problem. Most of these strategies divide the network's geographic area into sections called clusters. Each cluster has a cluster head (CH) responsible for managing local activities and communicating with other CHs. CHs are selected by parameters such as residual energy, location with respect to the sink node, coverage, and traffic load, among others. Clustering enhances energy efficiency since this technique switches off nodes inside the clusters while CHs take control [14], [15], [16]. Rotating the role of CHs among the sensors, can extend network life. However, these techniques use intricate mechanisms to select and rotate CHs. Recently, other research works [17], [18], [19], and [20] have addressed this energy concern by combining the A-star algorithm and fuzzy logic without clustering. Even if these mechanisms extend the network's lifespan, they do not always guarantee even energy depletion among sensor nodes. Other works use fuzzy strategies with ant colonies as routing algorithms to balance energy consumption [21], [22].

During the last few years, cellular automata models have shown to be suitable for model WSNs [9], [11]. In [6], for instance, the authors introduced a block CA approach to improve energy consumption in wireless sensor networks by selecting cluster heads in a geographical area. Although the energy consumption improves in a uniform deployment as the cluster size increases, their proposal is inefficient in random deployments. Moreover, the authors do not evaluate the energy balance among sensors. In [7], an AC algorithm is developed to achieve better coverage quality and more equitable energy distribution among the nodes. However, the authors do not evaluate a routing scheme under scheduling rule transitions. Although cellular automata can be used in wireless sensor networks, proposing local-simple rules poses significant challenges. These rules should balance energy consumption using energy-efficient routing schemes that enable large-scale sensor applications.

Recently, learning automata (LA) have become a great alternative to studying WSNs. A learning automaton (LA) is a technique in which the learning process is based on either rewards or penalties. This process makes the learning automaton reach an optimal value by interacting with the environment as time passes. For instance, in [8], the authors proposed a cellular learning automaton method to select CH based on nodes' energy for heterogeneous deployments. However, these scenarios may not be well suited for all applications. Moreover, simulations show that cluster formation depletes a considerable amount of energy. In contrast, in [23], the sensor network is divided into different hierarchy levels in which the cluster head selection depends on both the number of active and available nodes. However, this proposal is tied to a specific amount of energy in each group of sensors. Research in [24] used an LA model to minimize the number of hops in a routing scheme using *Voronoi* diagrams. Simulation results of this model indicate that its behavior improves when a larger number of nodes in the network is present. The proposal in [10] used an LA model to optimize a routing algorithm for a dynamic network environment. In [25], the authors claim that LA enhances energy consumption, energy balance, and packet delivery ratio when combined with other techniques.

On the other hand, in a many-to-one communications, sensors closer to the sink node deplete their batteries faster than other sensors since they relay more packets to the sink node. This characteristic can potentially create holes or disruptions in the data collection process. To solve this issue, researchers have proposed node distribution strategies in areas with a heavier load to balance the energy in the network. The authors in [4], for instance, proposed a non-uniform node distribution strategy modeled as concentric coronas in which the sink node is located in the inner one. To balance energy consumption, the authors changed the node's density in each corona in a geometric progression from the outer to the inner coronas. In addition, the authors proposed a routing algorithm named *q-switch* to find the shortest path to the sink node based on residual energy. However, according to the authors' simulations, the outer corona still retains 15% residual energy when the network dies (let's recall then that ideally all nodes must deplete their batteries simultaneously). Furthermore, the authors do not consider energy consumption while processing data and control messages. In addition, to achieve homogeneous energy consumption, sensor nodes should be located in a predefined deployment that is not always possible or does not always suit all sensor applications well.

In [26], the authors split the network into uniform slices to balance energy consumption by using two strategies called *inter-slice* and *intra-slice*. In the *inter-slide* strategy, sensor nodes vary their transmission power using a linear programming model based on their distance from the sink node. This strategy allows sensors closer to the sink node to conserve energy since they transmit using a lower power level, while those located further away can consume more energy since they are not used as relay nodes. On the other hand, the *intra-slice* strategy considers the residual energy in which sensors using low power levels forward their packets to sensors using higher energy levels to save energy. By combining both strategies, the authors proposed an energy balance transmission protocol called ETP. Simulation experiments claim that ETP outperforms energy consumption compared with cluster-based algorithms such as [27]. However, the authors do not consider hardware limitations that cannot always set the required transmission power levels selected by

the algorithm. Furthermore, they do not consider the cost of sending control messages to know which sensors have a higher energy levels in their *intra-slice* strategy.

Based on [4], the authors in [3] proposed three strategies to balance energy depletion on corona-based wireless sensor networks. These strategies seek the optimal position of sensor nodes to minimize the number of sensors in each corona. To achieve this end, the transmission power control technique is used so that outer coronas can cover extended areas rather than coronas closer to the sink node since sensors near the sink node can use a lower transmission power level to balance energy depletion. Moreover, the authors proposed a technique in which sensor nodes can be inactive for long periods to save energy. However, the authors did not consider that off-the-shelf radios have power limitations that cannot always set the desired transmission power level [1]. Moreover, balancing energy consumption is achieved only for a predefined network deployment that is not well suited for all applications.

In a many-to-one communication pattern, the innermost corona in corona-based wireless networks is not always the highest energy-consuming region. Thus, the authors in [5] proposed a solution to identify the critical region in many-to-one networks. The authors claim that their proposal balances energy consumption and improves network lifetime compared with [3] and [4].

Recently, autonomous robots have used cellular automaton models to plan a collision-free path in unknown or hostile environments [28], [29], [30], [31], [32]. These mechanisms provide advantages in many-to-one networks since energy-efficient routing schemes, such as the A-star algorithm, can be used in path planning.

We envision our proposal as a suitable algorithm that can be combined with path planning algorithms to optimize energy consumption in WSNs employing a cellular automaton model. The model proposed in this paper is a discrete dynamic model that considers different behaviors of WSNs through a set of rules combined with a variation of the A-star algorithm. In particular, a simple energy-aware routing method based on the Game of Life cellular automaton is proposed, which provides a homogeneous energy depletion while extending the network lifetime considering factors such as residual energy, number of active neighbors, and a sleep schedule. Moreover, the proposed model maintains the simplicity of cellular automata.

In order to evaluate the effectiveness of GoS, a comparison with [4], [5], [17], and [20], is presented in Section V. Contrary to these works, which use fuzzy logic and corona-based networks, the proposed method presents a straightforward cellular automaton model based on the Game of Life that intrinsically allows turning sensor nodes off/on. This characteristic enhances network lifetime while balancing energy consumption. Moreover, the A-star algorithm in GoS can create dynamic routes that can adapt in regular or irregular network deployments.

III. CELLULAR AUTOMATON

Cellular automaton (CA) can be viewed as a spatially extended decentralized system consisting of several individual entities (called cells). Cellular automata are mathematical models of dynamical systems that evolve in a parallel way in discrete time steps. Thus, at each interaction, the state of each cell is changed according to a set of local dynamic transition rules that take into account the cells' own state and the state of neighboring entities.

Formally, CA can be defined as a four-tuple, $\{\mathcal{C}, \Omega, V, f\}$; where $\mathcal{C}$ is the cellular space consisting of a set of individual entities. Ω denotes a finite set of states whose elements are all possible cell states. V denotes the cell neighborhood of each entity $\in \mathcal{C}$, and f denotes the local transition rules, which specify how CA evolves.

In particular, for a two-dimensional CA, cellular space $\mathcal{C}$ is represented as a regular spatial lattice or grid $\mathcal{C}$ of $L \times M$ cells, $\mathcal{C} = \{(i, j) \mid i, j \in \mathbb{Z}, 1 \leq i \leq L, 1 \leq j \leq M\}$. At time t, each entity stays in one of the finite numbers of possible discrete states in Ω. By interacting with the entities in its neighborhood V, each agent updates its current state following the set of specific transition rules in f. Thus, let $\omega_{i,j}(t)$ be the state of a cell $c = (i, j)$ at time t, which can be defined as follows:

$$\omega_{i,j}(t) = f(\omega_{i,j}(t-1), \{\omega_{k,l}(t-1)\}), (k, l) \in V_{i,j},$$

where f is an arbitrary function that specifies the cellular automaton rule operating on $V_{i,j}$, i.e., the set of cells (k, l) in the cell's neighborhood (i, j).

The cell's neighborhood can have different shapes and neighborhood radii. However, the two most popular neighborhoods in CA are the Moore and von Neumann neighborhoods. Figures 1 and 2 show the Moore and von Neumann neighborhoods with different radii on a two-dimensional square lattice, respectively. The focal cell is colored black, while its corresponding neighborhood is colored gray.

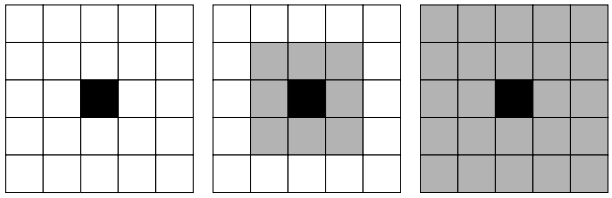
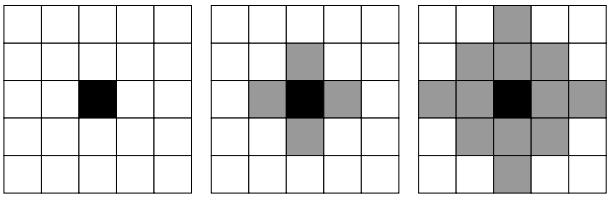


FIGURE 1. Moore neighborhood.

A. THE GAME OF LIFE

A well-known example of cellular automata is the Game of Life (also known as GoL). This game has no players, i.e., its evolution is determined by its initial state and therefore does not require any input from the player [33], [34]. As a result, interaction with the GoL is achieved by creating an initial configuration and observing how it evolves over time. Recently, the Game of Life cellular automaton has been used



(a) $r = 0$ (b) $r = 1$ (c) $r = 2$

FIGURE 2. Von Neumann neighborhood.

to solve different problems [35], [36], [37]. In the GoL CA, each cell can take two states (alive or dead), generally represented as 1 and 0, respectively. The cells' states are updated simultaneously and in discrete time. In the two-dimensional neighborhood, the local transition rules of the GoL are as follows:

$$\omega_{i,j}(t) = \begin{cases} 1, & \omega_{i,j}(t-1) = 1\,, \sum\limits_{(k,l)\in V_{i,j}} \omega_{k,l}(t-1) = 2, 3 \\ 0, & \omega_{i,j}(t-1) = 1\,, \sum\limits_{(k,l)\in V_{i,j}} \omega_{k,l}(t-1) \neq 2, 3 \\ 1, & \omega_{i,j}(t-1) = 0\,, \sum\limits_{(k,l)\in V_{i,j}} \omega_{k,l}(t-1)) = 3 \\ 0, & \omega_{i,j}(t-1) = 0\,, \sum\limits_{(k,l)\in V_{i,j}} \omega_{k,l}(t-1)) \neq 3, \end{cases} \tag{1}$$

where $\sum\limits_{(k,l)\in V_{i,j}} \omega_{k,l}(t-1)$ denotes the number of active (alive) cells at time-step $t-1$, in the neighborhood $V_{i,j}$ of the focal cell (i, j).

B. GoL FOR WSNs

Although the Game of Life cellular automaton emerged as a biological model, this CA can be used in WSNs. For instance, in WSNs, nearby sensors usually gather similar data measurements. Thus the above transition rules avoid too many sensors sending redundant information to the sink node since these rules turn off the majority of nearby sensors at each step. Moreover, these rules also save energy since only two or three sensors can be active at a time in the surroundings of a focal cell (hereafter the words cell and sensor will be used interchangeably). In order to use GoL as a WSN model, it's necessary to map the concepts of CA to WSNs components, where C represents the entire network composed of sensors. Ω is the set of possible states for each sensor (active and inactive). V denotes all the sensors within the focal sensor's communication range, and f designates the algorithms and protocols (rules) that each sensor must follow. Table 1 summarizes the node attributes.

Algorithm 1 shows how the GoL automaton is used in a WSN.

Enhancing network lifetime is one of the main concerns in WSNs since sensors are usually equipped with small batteries that cannot be straightforwardly recharged or replaced. Therefore, applying energy-efficient protocols has become a priority topic. For instance, recently, a long-term pattern known as *symmetric die-hard* cellular automaton [38], which lasts 1638 ticks in a 32 × 32 bounding box was proposed. Figure 3 shows this CA at different time-steps. Even if this model can extend lifespan in a two-dimensional lattice, it creates patterns shown in Figs. 3b-3f that do not fit well in WSNs since a large number of sensors remain switched off for several time-steps, causing a loss of information. Moreover, routing protocols may not work properly with these rules since no nearby relay sensors exist. It is thus necessary to introduce variations to these rules to allow sensors to turn on when they have been inactive for many time-steps.

TABLE 1. Attributes for a sensor (i, j) at time t.

Attribute	Description
$\omega_{i,j}(t-1)$	Current state of the cell at time $t-1$
$\omega_{i,j}(t)$	State to which the cell will transit to at time-step t
$e_{i,j}$	Battery charge of the cell (i, j)

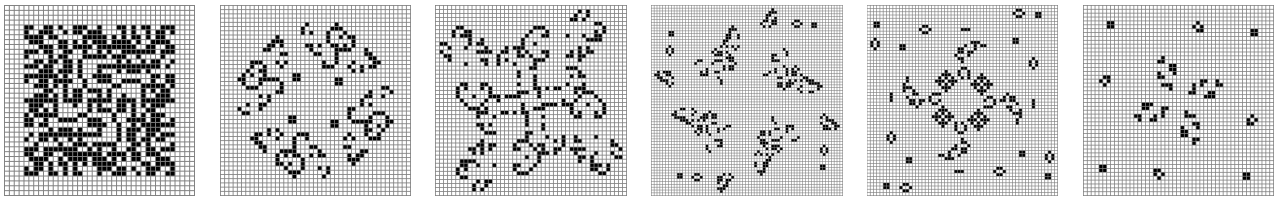
Algorithm 1 Game of Life Algorithm

Input: current_node
Output: null
Check the node's neighborhood to apply rules:

```
active_neighbors = 0
for node in current_node.neighborhood do
    if node.status == active then
        active_neighbors + = 1
    end if
end for
if current_node.status == active then
    if active_neighbors == 2 or 3 then
        current_node.next_status = active
    else
        current_node.next_status = inactive
    end if
end if
if current_node.status == inactive then
    if active_neighbors == 3 then
        current_node.next_status = active
    else
        current_node.next_status = inactive
    end if
end if
return null
```

IV. GAME OF SENSORS

This section presents a new set of transition rules added to the Game of Life model to extend the network lifetime while balancing energy consumption among sensors. Then, since these new rules cannot relay packets, the A-star algorithm is adapted to fulfill this role. The fusion between the adapted A-star algorithm and the new rules of the GoL is therefore called the Game of Sensors (GoS). Finally, an evaluation of



(a) $t = 1$ (b) $t = 42$ (c) $t = 76$ (d) $t = 851$ (e) $t = 1112$ (f) $t = 1480$

FIGURE 3. Symmetric die hard pattern.

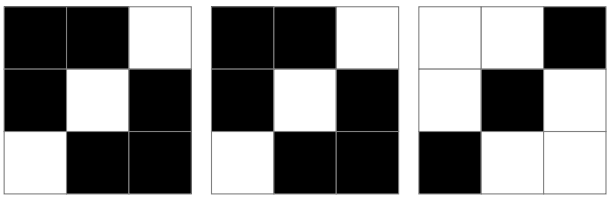
the new set of rules is presented to measure the effectiveness of the proposed method.

A. TRANSITION RULES IN THE GoS MODEL

In WSNs, keeping devices continuously active for long periods quickly depletes their battery. On the contrary, if a sensor remains inactive for extended periods, a loss of information may be caused since there might not be enough awake sensors to relay data packets to the sink node. Figures 3b-3f show this scenario, in which there are many areas of cells that are inactive, resulting in a loss of information. In order to solve this situation, the following rules are added as a solution to the GoL model for WSNs:

$$\omega_{i,j}(t) = \begin{cases} 1, & \sum_{z=1}^{RT} \omega_{i,j}(t-z) = 0 \\ 0, & \sum_{z=1}^{RT} \omega_{i,j}(t-z) = RT, \end{cases} \tag{2}$$

where RT denotes the number of time-steps that a sensor remains in the same state. Therefore, these transition rules assure that if a sensor has remained active for RT time-steps ($\sum_{z=1}^{RT} \omega_{i,j}(t-z) = RT$), it will change its state to rest ($\omega_{i,j}(t) = 0$). On the contrary, if a sensor has remained resting for RT time-steps ($\sum_{z=1}^{RT} \omega_{i,j}(t-z) = 0$), it will become alive ($\omega_{i,j}(t) = 1$). In both cases, the focal cell's state will change, regardless of its neighborhood's state. From this point onward, these rules will be identified as the sleep schedule rule. Figures 4a and 4b show a set of cells following the same pattern (from $t = 1$ to $t = 10$, i.e., $RT = 10$), whereas Fig. 4c shows how (2) breaks the pattern.



(a) $t = 1$ (b) $t = 10$ (c) $t = 11$

FIGURE 4. Sleep schedule rule.

Algorithm 2 describes the implementation of the sleep schedule rule in a sensor node.

Algorithm 2 Sleep Schedule Algorithm

Input: node
Output: null
Check the node's activity to apply the sleep schedule:
node.next_status = node.next_status from algorithm 1
if node.status == active **then**
  node.battery -= 1
  node.active_time += 1
  node.inactive_time = 0
else
  node.active_time = 0
  node.inactive_time += 1
end if
if node.active_time $\geq RT$ **then**
  node.next_status = inactive
end if
if node.inactive_time $\geq RT$ **then**
  node.next_status = active
end if
return null

In addition, the foremost concern in WSNs is the depletion of batteries. Thus, cells with a high battery charge should remain active with a high probability, whereas cells with a lower battery charge should rest to save energy. Then, if the current energy e of a sensor is below a threshold th and its next state is active ($\omega_{i,j}(t) = 1$)), it will change its state to active with probability p. This rule produces that sensors with low energy switch to active state less frequently. Therefore, this rule produces that sensors with low energy switch to active state less frequently. This rule is implemented in Algorithm 3. Table 2 contains a summary of the parameters of the proposed model.

TABLE 2. Parameters of the proposed model.

Parameter	Description
RT	Sleep schedule, which denotes the number of time steps that a sensor remains in the same state
th	Energy threshold of a sensor to change its state to active
p	Probability that a sensor change its state as a function of its battery charge

B. A MODIFIED A-STAR ALGORITHM

Since the proposed rules do not relay packets, the A-star algorithm is adapted to carry out the routing process [39].

Algorithm 3 Threshold Algorithm

Input: node
Output: null
Check the node's battery to apply the rules:
node.next_status = node.next_status from algorithm 2
if node.next_status == active **then**
  if node.battery < *th* **then**
    r = random number from 0 to 1
    if r > *p* **then**
      node.next_status = inactive
    end if
  end if
end if
return null

A-star is a widely known routing algorithm using heuristic knowledge to select the optimal path from source to destination. To achieve this end, the A-star algorithm uses the following two-part function to evaluate the best relay node to reach the destination:

$$f(n) = g(n) + h(n), \tag{3}$$

where $g(n)$ is the actual cost from the start node to the current node n, while $h(n)$ is a heuristic function that estimates the cost from the current node n to the destination node (the sink node). A-star maintains two priority queues: an open list and a closed list. The open list contains those nodes that have been evaluated but have not been selected as relay nodes, whereas the closed list contains those that have already been examined. Initially, the start sensor is the only node in the open list. Then, the algorithm checks if this sensor is the destination node. If so, the task is complete. Otherwise, the algorithm evaluates the cost required to extend the path through using its neighborhood (see (3)) to choose the best relay node. In this way, the A-star algorithm finds an optimal route in which obstacles and nodes do not change over time. However, in the proposed model, sensors may change their state every time-step. Thus, it is necessary to implement a modified version of the A-star algorithm.

Since in the proposed method, an active cell cannot estimate which neighbors wake up or maintain an active state in the next time-step since each cell depends on the behavior of its surroundings expressed by Algorithms 1-3. The adapted A-star algorithm is applied as follows: component $g(n)$ is calculated by computing the Euclidean distance from the current cell position n to the focal active neighbor m. In contrast, $h(n)$ can use any estimator from the focal neighbor to the destination node. For the sake of simplicity, this paper uses, Euclidean distance. However, any other estimator can be used, such as Manhattan distance, Minkowski distance, or the estimator used in [17], for instance. In this way, this version of the algorithm computes the best relay node in each time-step.

For cases in which (3) applies to the current cell n, and it cannot look for a better relay cell, the current cell n has to wait for the next time-step to look for a better relay cell. This situation can occur because variations in local transition rules change the possible relay nodes at each time-step over time, provoking that the active neighbors at time t do not enhance the distance from the current node to the destination node. Due to this situation, the rule described in the equation 4 is added to the proposed method.

$$\omega_{i,j}(t) = \begin{cases} 1, & \omega_{i,j}(t-1) = 1,\ h(n) > d(n, DST) \\ 0, & \omega_{i,j}(t-1) = 1,\ h(n) < d(n, DST), \end{cases} \tag{4}$$

where $d(n, DST)$ represents the Euclidean distance from the current node n to the destination node. This rule ensures that the distance to the destination node diminishes every time a packet is sent. If the current node n cannot look for a better relay at time $(t-1)$, it will remain active in the next time-step to again search for a better relay. Otherwise, the node can send the packet, thus becoming inactive. Algorithm 4 describes the adapted routing algorithm.

Algorithm 4 Routing Algorithm

Input: current_node
Output: next_node
Check the node's neighborhood to choose the next jump
current_distance = distance from current_node to the sink
current_f = highest possible value
for node in current_node.neighborhood **do**
  if node.status == active **then**
    f_node = g_node + h_node
    if f_node < current_f **then**
      current_f = f_node
      node.distance = distance from node to the sink
      if node.distance < current_distance **then**
        current_distance = node.distance
        next_node = node
      end if
    end if
  end if
end for
if next_node == sink **then**
  Packet received
end if
return next_node

As can be observed from Algorithm 4, the complexity of this modified version is $O(n)$, where n is the number of neighbors of the transmitting sensor.

C. GoS ALGORITHM

Based on Algorithms 1–4, the complete GoS algorithm is presented in Algorithm 5.

D. PARAMETER TUNING

In this section, we first study the main parameters of the GoS method. According to the design goals, our proposal should balance the energy in the network, i.e., prevent sensors close to the sink deplete their batteries quickly. We then compare

Algorithm 5 Game of Sensors Algorithm

Input: network
Output: null
Execute the proposed algorithm:
Set an initial scenario
while all nodes in network are alive **do**
  if No packet in network **then**
    Generate packet in a random node
  end if
  for node in network **do**
    Game of life algorithm (node)
    Sleep schedule algorithm (node)
    Threshold algorithm (node)
    Routing algorithm (node)
  end for
  for node in network **do**
    node.next_status = node.next_status from previous algorithms
    node.status = node.next_status
  end for
end while
return null

our proposal with the Game of Life-based CA to assess the impact of the new transition rules. For this evaluation, the routing process is not applied since the GoL has no such functionality (It will later be included; in Section V). A custom simulator written in Python language is used. The studied region corresponds to a regular two-dimensional square lattice of 30×30 cells. For the sake of simplicity, each cell has a battery of 100 units of charge. Each active time-step consumes one unit of battery. Each time-step corresponds to one second. In Section V, we will make use of a more realistic energy model to compare the GoS method with the most relevant related works. The energy threshold th is set to 50%, while the probability of change is set to $p = 0.6$. The impact of choosing the value of p will be explored later. The initial state of the CA is determined randomly, in which each cell has a 50% probability of being active and 50% of being inactive. Each metric considered in the simulation was run 100 times to obtain average results.

In the new set of rules, variations in the sleep schedule (RT) imply changes in the number of active sensors. Figure 5a shows how RT changes the average number of active cells in the network. For instance, when RT is equal to 4, the average active cells is 31.8%. In other words, 68.2% of cells are resting. Conversely, when RT is equal to 20, the average number of active cells is 22.3%. Figure 5b, on the other hand, shows how RT affects network lifetime. In this figure, for RT equal to 2, the network lifespan in time-steps is 189, while for RT equal to 20, the network lifespan increases to 271, i.e., the network lifetime increases by 30%. In this way, by varying the value of RT, the network lifetime and the average number of active cells can shrink or extend gradually depending on which value of RT is selected. Thus, depending on the parameter of interest (active cells or lifespan), RT should be selected accordingly. It is relevant to note that each simulation ends when the first cell depletes its battery.

It is relevant to note in Figure 5a that the average number of active cells first increases for $RT < 5$ and then decreases for $RT > 5$. This situation occurs since small values of RT cause drastic changes in the state of sensors, i.e., sensors are forced to be active/inactive in a shorter period. At the same time, the GoL rules will drastically reduce the average number of active cells every time the RT is satisfied. This situation provokes the network works in an unstable mode since some sensors are active for more time than others. As a result, one of these sensors depletes its battery faster, provoking that the simulation stops untimely. However, for $RT > 5$, the network reaches a stable state, i.e., the number of active sensors remains similar as time passes since sensors change their state less frequently, incrementing the network lifetime.

In GoS, RT is set to 10 since this value establishes a fair point between the network lifetime and the number of active cells to guarantee that routing protocols work correctly, as shown later.

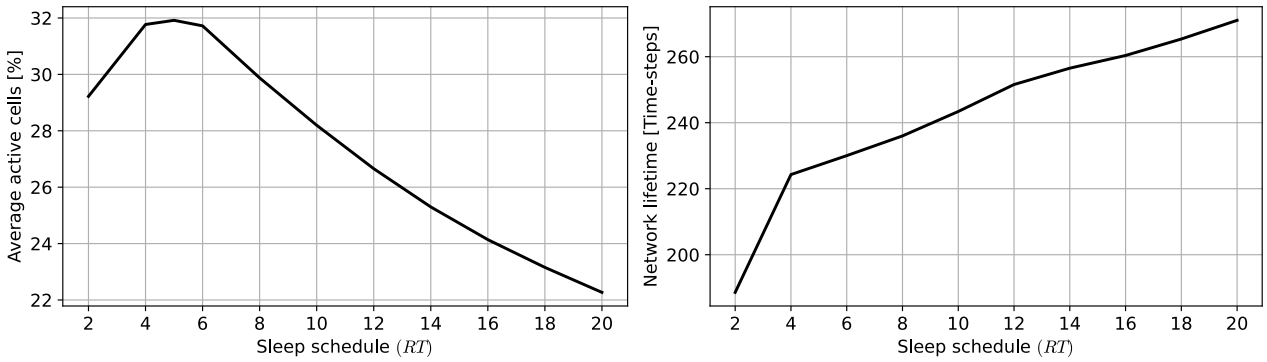
Once all these parameters are set ($th = 50\%$, $p = 0.6$, and $RT = 10$), Fig. 6 shows the number of active cells over time. At the beginning of the simulation, as aforementioned, 50% of cells were active due to the initial conditions. After that, there is a brief period of oscillation while the system converges at 28% of active cells. This figure also shows that when the first sensor runs out of battery, the number of active cells remains constant from time-step 10 to time-step 242. After that, the average active cells diminishes quickly until all cells discharge their batteries.

At time-step 350 (see Fig. 6), only 11% of the active sensors can be used in routing protocols. This situation may result in a loss of information since there are few relay sensors available to transmit data. However, the number of active cells can be extended by reducing RT at the cost of reducing the network lifespan.

Figure 7a shows the network state when the first sensor runs out of battery. This sensor is marked in the lower right of the figure ($\times$). Black squares denote the active cells at this point, whereas white squares denote the resting cells. Figure 7b shows the network state when 20% of sensors run out of battery (see gray cells).

One of the most relevant parameters to consider in this work is the number of active neighbors per sensor that allow routing protocols to operate correctly. Figure 8 shows the average number of active neighbors after 1% of the cells run out of battery. It can be observed in this figure that for the one-hop neighborhood, there are at least two relay neighbors per sensor until 20% of the sensors die, whereas there are at least six neighbors per cell for the two-hop neighborhood. This figure also shows that after 20% of the sensors run out of battery, the remaining active neighbors decrease rapidly.

Figure 9 shows the probability of change for different values of p. This figure shows that as long as p increases,



(a) Average active cells vs. sleep schedule (RT)

(b) Network lifespan vs. sleep schedule (RT)

FIGURE 5. Sleep schedule variation with respect to active cells and network lifespan. For both figures *th* is set to 50%, while $p = 0.6$.

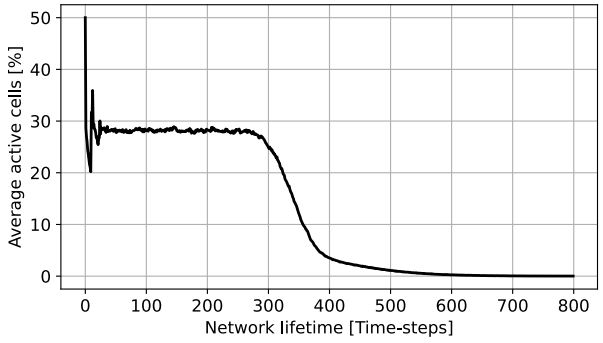
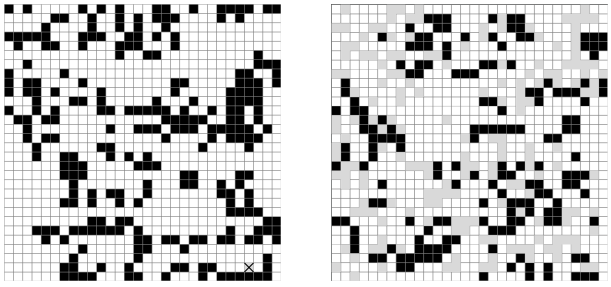


FIGURE 6. Average active cells vs. network lifetime.



(a) Network state when the first nodes die

(b) Network state when 20% of the nodes die

FIGURE 7. Network state at different times.

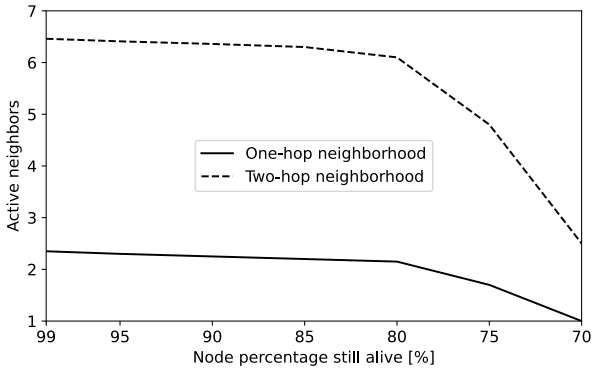
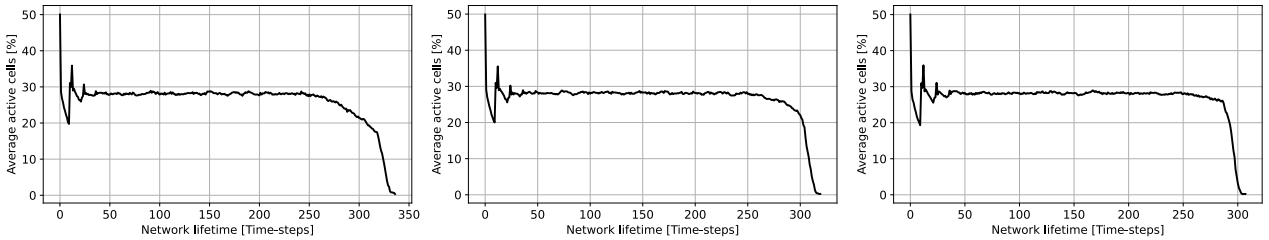


FIGURE 8. Active neighbors vs. nodes still alive.

the network lifespan decreases. For instance, for $p = 0.4$, the network dies at 335 time-steps, while for $p = 0.8$, it dies at 307 time-steps. This figure also shows that for $p = 0.8$, the number of active sensors diminishes quickly after the first node runs out of battery (at time-step 250) compared to when $p = 0.4$. This situation occurs because when *th* is below 50%, the probability of change forces the rest of the sensors to turn on fewer times, resulting in slower battery discharge. In GoS, p is selected as 0.6 since this value establishes a fair point in maintaining network lifetime while keeping 28% of network coverage.

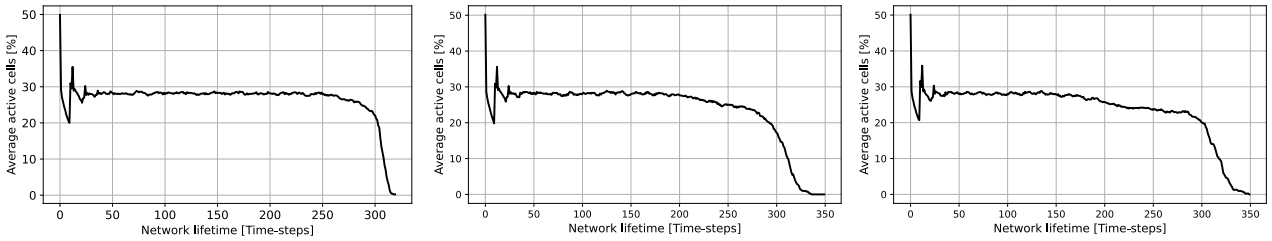
Figure 10 shows the threshold *th* when $p = 0.6$. This figure shows that the larger the value of *th*, the faster the average number of active nodes decreases. For instance, for $th = 50\%$, the average active cells start to decrease at 250 time-steps, while for $th = 70\%$, the average active cells start to decrease after 150 time-steps. This result compromises the routing protocol since fewer active nodes are less likely to find a route to the destination node. In GoS, *th* is selected as 50% since this value establishes a fair point in the average number of active sensors while extending network lifetime.

The following experiment was conducted to evaluate the Game of Life CA's effectiveness. The same regular two-dimensional square lattice of 30 × 30 cells was used. Each cell has a battery with 100 units of charge. One unit of battery is consumed for each active time-step. Each time-step corresponds to one second. The initial state of the GoL is determined randomly, each cell having a 50% probability of being active. 100 simulations were run to obtain average results. Table 3 shows the average network lifetime, the average number of active cells, and the average number of neighbors in the GoL CA. This table demonstrates that the standard deviation in the network lifetime field is 10.66 times higher compared to the standard deviation in the network lifetime field of the proposed model (see Table 4). This result shows that GoL cannot maintain the same number of active cells as time passes compared to GoS. This effect can be seen in Fig. 6, in which the number of active cells remains



(a) $p = 0.4$ (b) $p = 0.6$ (c) $p = 0.8$

FIGURE 9. Variations of probability of change (*p*).



(a) $th = 50\%$ (b) $th = 60\%$ (c) $th = 70\%$

FIGURE 10. Variations of threshold (*th*) with $p = 0.6$.

TABLE 3. Game of life CA effectiveness.

Game of Life	Average	Standard deviation
Network lifetime	283.18	95.24
Active cells	6.97	1.22
One-hop neighborhood	0.54	0.09
Two-hop neighborhood	1.63	0.29

TABLE 4. GoS model effectiveness.

Game of Sensors	Average	Standard deviation
Network lifetime	241.6	8.93
Active cells	28.2	0.3
One-hop neighborhood	2.2	0.02
Two-hop neighborhood	6.5	0.07

constant until 20% of cells run out of battery. Moreover, the proposed technique maintains about 28% of cells active while the Game of Life has only 6.97%. In addition, the GoS method has about two relay neighbors in the one-hop neighborhood, whereas the Game of Life has only 0.54. The latter result impacts routing protocols since, most of the time, the GoL CA does not have relay neighbors. Finally, the GoS method for the two-hop neighborhood has at least six relay neighbors, while the GoL CA has only 1.63. It is relevant to note that Tables 3 and 4 report average results until the first node runs out of battery.

Table 5 shows the proposed technique's average number of time-steps, average coverage, and average number of neighbors when 20% of the sensors run out of battery. It can be seen that the proposed method maintains a good energy balance and enough relay sensors for routing protocols to operate.

TABLE 5. GoS model effectiveness when 20% of cells run out of battery.

Game of Sensors	Average	Standard deviation
Network lifetime	322.2	3.64
Active cells	27.58	0.29
One-hop neighborhood	2.01	0.02
Two-hop neighborhood	6.08	0.06

V. EXPERIMENTS AND RESULTS

This section presents a performance evaluation of the GoS model. For this, simulation experiments were divided into three sets. First, a comparison with the EERP [17] and A&F [40] algorithms is carried out. Then, a comparison with the LPA-star algorithm [20] is presented. Finally, a comparison with [5] is conducted. Hereafter, this algorithm will be identified as "Maximizing." It is important to mention that the Maximizing proposal outperforms [3] and [4]. A custom simulator in Python was used for these three sets of experiments. The following assumptions and properties are presented in the three sets of experiments.

- All sensor nodes have the same initial energy except the last experiment.
- All sensor nodes have the same maximum transmission range.
- Each sensor node is synchronized with its one-hop neighborhood.
- Each sensor node knows the location of the sink node as well as the location of its neighbors.
- All sensor nodes are located in the same position during the simulation.
- The model evolves every time-step.

A time-step represents the time required to evolve the system, defined as $C + Hello + Unicast + Packet$, where

C denotes the contention period (MAC layer) to send *hello* packets. The *Hello* period represents the time to send *hello* packets. This time is required to announce which nodes are active during the current time-step. It is relevant to mention that during the *Hello* period, all sensor neighbors should receive *hello* messages to update the cell state at time $t+1$. Based on Fig. 8, the average number of active neighbors in the one-hop neighborhood scenario is approximately two for each time-step in regular deployments. This feature means that, on average, only two sensor nodes try to gain the channel (C period) and send *hello* messages every time-step. Table 4 confirms this value (the one-hop neighborhood field) with a standard deviation equal to 0.02. Finally, the *Packet* period is the time in which to send a packet. It is relevant to say that packets are sent in a unicast manner by means of an RTS-CTS protocol or similar techniques. Thus, only the sensor receiving the message should be active, whereas the rest can switch off until the next time-step.

For these three sets of experiments, in GoS, RT is set to 10, p is set to 0.6, and th is set to 50%.

A. ENERGY MODEL

The radio energy consumption model was taken from [5] and [17], in which the energy consumed by transmitting l bits can be calculated as follows:

$$E_{Tx}(l,d) = \begin{cases} l \cdot E_{elec} + l \cdot \epsilon_{fs} \cdot d^2 & \text{if } d \leq d_0 \\ l \cdot E_{elec} + l \cdot \epsilon_{amp} \cdot d^4 & \text{if } d > d_0, \end{cases} \quad (5)$$

where l represents the packet size, while d is the Euclidean distance between the transmitter and the receiver. E_{elec} represents the transmitter and receiver circuit dissipation per bit, ϵ_{fs} represents the amplification factor for the free space model (path loss equal to 2), while ϵ_{amp} denotes the amplification factor for the multi-path model (path loss equal to 4). Finally, $d_0 = \sqrt{\frac{\epsilon_{fs}}{\epsilon_{amp}}}$.

The energy consumption model when receiving an l-bits message is calculated as follows:

$$E_{Rx}(l) = l \cdot E_{elec}. \quad (6)$$

Since proposals in [3], [4], and [5] do not consider the cost of the energy consumption when processing data, the proposed model in [15] is taken as follows:

$$E_{Tx}(l,d) = \begin{cases} l \cdot E_{elec} + l \cdot E_{DA} + l \cdot \epsilon_{fs} \cdot d^2 & \text{if } d \leq d_0 \\ l \cdot E_{elec} + l \cdot E_{DA} + l \cdot \epsilon_{amp} \cdot d^4 & \text{if } d > d_0, \end{cases} \quad (7)$$

where E_{DA} denotes the energy consumption when processing a data packet, while the energy consumption model when receiving and processing an l-bits message is calculated as follows:

$$E_{Rx}(l) = l \cdot E_{elec} + l \cdot E_{DA}. \quad (8)$$

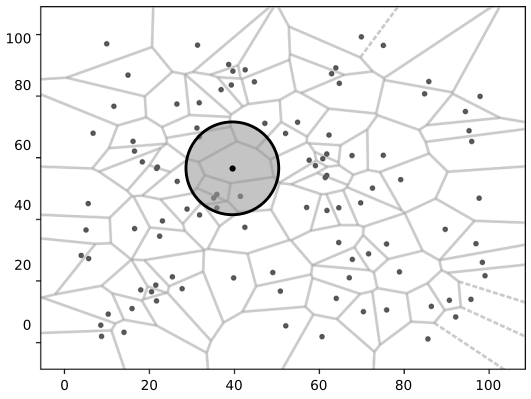


FIGURE 11. A 2D randomly distributed network measuring 100 m × 100 m.

B. GoS VS. EERP

For this set of experiments, the studied region corresponds to a 2D square area measuring 200 m × 200 m. Although GoS was modeled in a regular arrangement (see Section IV), it can also be applied to irregular deployments, as shown in Fig. 11, in which the focal cell is depicted in a 2D region of 100 m × 100 m. However, instead of using the Moore neighborhood (8 neighbors), the number of active sensors is computed as a percentage ($\frac{2}{8} = 0.25$, and $\frac{3}{8} = .375$). In other words, the focal sensor should calculate the number of active sensors in its vicinity as follows: if the current state of the focal cell is active; and the percentage of the active sensors in its neighborhood is between 25% and 37.5%, the focal sensor remains active. On the other hand, if the current state of the focal cell is inactive, and the percentage of the active cells in its vicinity is 37.5%, then the focal cell will change to active. However, in an irregular arrangement, it is not always possible to have a neighborhood with exactly 37.5% of active cells; thus, a range between 35% and 40% will be used instead. These rules follow a similar proportion as proposed in (1). Figure 12 shows the average number of active cells vs. the network lifetime in an irregular arrangement corresponding to a 2D square area measuring 100 m × 100 m. The cell's radius is set to 15 m. One hundred sensors were randomly deployed. These simulations were run 100 times to obtain average results. For this set of experiments, the average number of the one-hop neighborhood is 2.1, whereas the percentage of active cells until the first node runs out of battery is 24%. When comparing this to the values shown in Table 4 for $p = 0.6$ (regular arrangement), both values diminished slightly. However, regular and irregular arrangements have shown similar behavior (see Figs. 6 and 12).

Table 6 contains a summary of the parameters used for this set of experiments. Before explaining the results, it should be noted that the EERP algorithm has twenty actor nodes that generate data. These nodes randomly change every 100 rounds in these simulations. It is also relevant to note that control messages (*hello* and *unicast*) in the GoS model should be considered since the neighborhood of a focal sensor

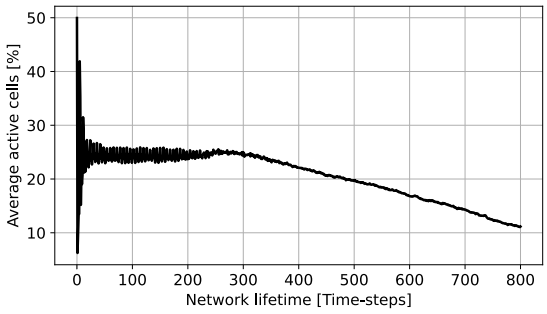


FIGURE 12. Number of active cells vs. network lifetime in an irregular arrangement. For this figure *th* is set to 50%, $RT = 10$, and $p = 0.6$.

TABLE 6. Simulation parameters.

Parameter name	Value
Number of nodes (N)	50
Transmission radio range	80 m
Packet length (l)	4000 bits
Initial Energy (E_0)	5 J
Circuit dissipation (E_{elec})	50 nJ/bit
Free-space amplifier(ϵ_{fs})	100 pJ/bit/m^2
Data processing factor (E_{DA})	5 nJ/bit
Number of actors (A)	20
Radio signal range of actors	30 m
Control packet size	200 bits
Studied region	200 m $\times$ 200 m
Sink position	(200 m, 200 m)

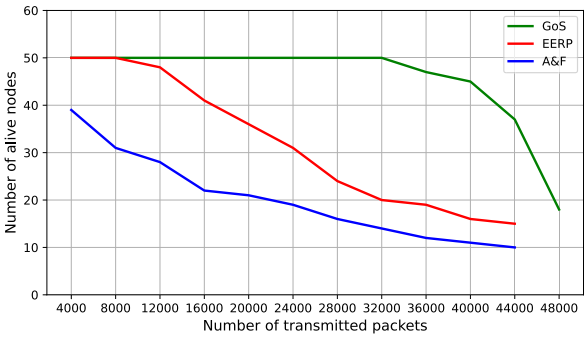


FIGURE 13. Number of alive nodes vs. number of transmitted packets comparing GoS, EERP, and A&F.

requires knowing the number of active sensors during the time-step to update the next time-step. For this purpose, (7) and (8) are used.

Figure 13 shows the number of transmitted packets vs. the number of alive nodes. It can be seen that GoS outperforms the number of transmitted packets compared to the EERP and A&F methods. This is a consequence of the GoS simplicity since it intrinsically manages active and inactive periods and the sleep schedule (RT). Moreover, the first node dies after 32, 000 transmitted packets in the GoS model, while for EERP, it dies at 8, 100. In other words, GoS can triple the number of transmitted packets. This figure also shows that after the first node dies, the number of alive nodes diminishes slowly in the GoS method compared to A&F and the EERP schemes. This behavior can also be seen in Fig. 12, in which

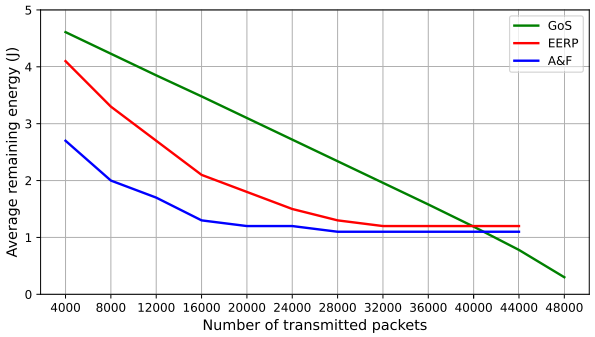


FIGURE 14. Average remaining energy vs. number of transmitted packets comparing EERP, A&F, and GoS method.

TABLE 7. Simulation parameters.

Parameter name	Value
Number of nodes (N)	50
Transmission radio range	80 m
Packet length (l)	4000 bits
Initial Energy (E_0)	5 J
Circuit dissipation (E_{elec})	50 nJ/bit
Free-space amplifier(ϵ_{fs})	100 pJ/bit/m^2
Data processing factor (E_{DA})	5 nJ/bit
Control packet size	200 bits
Studied region	200 m $\times$ 200 m
Sink position	(200 m, 200 m)

the number of alive cells does not fall instantly after the first node runs out of energy, resulting in a longer lifespan (transmitted packets). This feature is due to choosing $p = 0.6$. It should be noted that even when the authors in [17] range the transmitted packets from 4, 000 to 44, 000, our simulations continue until no more packets can reach the sink node (48, 000 packets).

Figure 14 shows how the average remaining energy in A&F, EERP, and GoS decreases as the number of transmitted packets increases. It can be seen in this figure that GoS diminishes energy availability in senor nodes homogeneously compared to EERP and A&F.

C. GoS VS. LPA-STAR

LPA-star takes advantage of reusing previous searches in the A-star algorithm to seek a better path. The use of this mechanism enhances performance compared with the EERP algorithm. For this set of experiments, the studied region corresponds to a 2D square area measuring 200 m $\times$ 200 m. Fifty sensor nodes were placed randomly in the studied region to create a similar scheme presented in LPA-star [20]. Simulation experiments were run 100 times to obtain average results. For each experiment, the sink node in the proposed model was fixed inside the studied region at coordinates (200 m, 200 m). It is relevant to note that for this set of experiments, (7) and (8) were used, even when authors in [20] and [17] do not include control processing consumption in their experiments. The authors in [20] range packets from 1 to 20, 000. Table 7 contains a summary of the parameters used for this set of experiments.

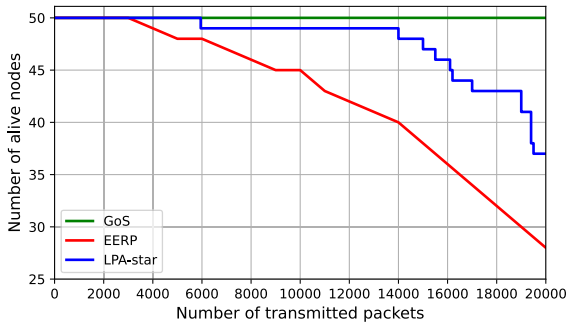


FIGURE 15. Number of alive nodes vs. number of transmitted packets comparing EERP, LPA-star, and GoS method.

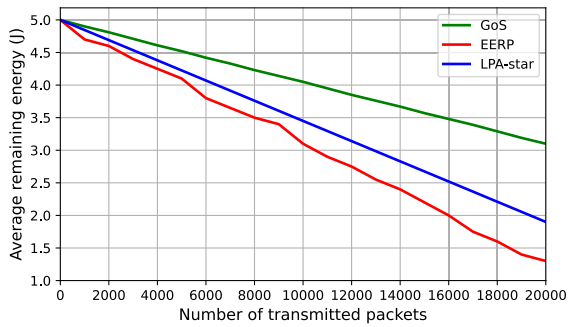


FIGURE 16. Average remaining energy vs. network lifetime comparing LPA-star, EERP, and GoS method.

Figure 15 illustrates how GoS outperforms the number of transmitted packets (network lifetime) compared to LPA-star and EERP. This figure also shows that the GoS method maintains the number of alive nodes as the number of transmitted packets increases compared to LPA-star and EERP before the first node dies. This behavior is a consequence of the probability of change, and the sleep schedule selected, see Section IV-D. For instance, the first node dies in EERP after 2, 947 packets, while in LPA-star occurs after 5, 995 packets. However, in GoS, the first node dies after 24, 779 packets. Figure 16 shows how the average remaining energy in LPA-star, EERP, and GoS decreases as the number of transmitted packets increases. It can be seen in this figure that LPA-star and GoS diminish their energy resources homogeneously compared to EERP. However, GoS shows a better energy balance.

D. GoS VS. MAXIMIZING

Since the authors in the Maximizing protocol [5] and *q-switch* [4] do not consider the energy cost of control messages and data processing, a new implementation of the *q-switch* and Maximizing is used. Then, a comparison with GoS can be made since our proposal considers energy consumption in data processing and control messages.

For the *q-switch* and Maximizing algorithms, the studied region corresponds to a circular area divided into k concentric coronas, in which the sink node is located at the center of the innermost corona. Specifically, in this set of experiments, $k = 5$, each corona has a width of 40 m. All sensors have the same maximum transmission range (80 m). The *ith* corona is denoted as C_i, and nodes in C_i can communicate directly with C_{i-1} and C_{i+1}. Nodes in the outermost corona do not forward any data. The number of nodes in each corona increases in a geometric progression of ratio equal to two from the outermost to the innermost coronas, as follows 3, 3, 6, 12, and 24 (see *q-switch* method [4]).

TABLE 8. Simulation parameters.

Parameter name	Value
Number of nodes (N)	48
Transmission radio range	80 m
Packet length (l)	4000 bits
Total Energy (T_E)	240 J
Initial Energy (E_0)	5 J
Circuit dissipation (E_{elec})	50 nJ/bit
Free-space amplifier(ϵ_{fs})	100 pJ/bit/m^2
Data processing factor (E_{DA})	5 nJ/bit
Control packet size	200 bits
Width of each corona	40 m
Radius of the network	$R = 200$ m
Sink position	(0 m, 0 m)

As mentioned in Section II, in a many-to-one transmission pattern, nodes close to the sink node deplete their battery faster than far away sensors. However, the innermost corona may not always be the most energy-consuming region. Thus, the authors in [5] proposed an equation to seek the corona with the highest energy dissipation rate. This strategy allows the network to balance the energy consumption by adjusting each sensor's initial energy so that all coronas run out of battery simultaneously. In other words, the network lifetime ends as soon as the first node dies since, at that moment, the rest of the nodes have zero remaining energy. Table 8 contains a summary of the parameters used for this set of experiments. Each experiment's initial amount of energy per node is distributed depending on corona location. Based on [5] and adding the energy cost to send control messages and data processing, the amount of energy per node from the outermost to the innermost coronas, is as follows: 4.258 J, 5.398 J, 5.6845 J, 5.6825 J, and 4.5325 J.

For the GoS model experiment, the studied region corresponds to a circular area having a radius equal to 200 m. Forty-eight sensor nodes were placed randomly in the studied region, in which the sink node is located at the center of the area. The initial energy for each node is 5 J. Table 8 contains a summary of the parameters used for this experiment. Figure 17 shows how GoS outperforms the *q-switch* and the Maximizing algorithms in terms of the number of transmitted packets (network lifetime). Even when the first node in the GoS method dies before compared to *q-switch* and Maximizing, GoS can transmit more packets. For instance, the Maximizing model transmits 34, 123 packets, whereas GoS can transmit 47, 977 packets. In other words, the GoS technique improves the network lifetime by 1.4 times compared to the Maximizing algorithm and 1.9 times compared to the *q-switch*.

Although the *q-switch* and Maximizing methods achieve a better energy balance, the GoS method can transmit more

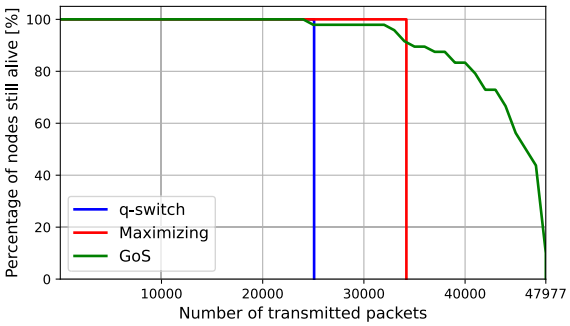


FIGURE 17. Percentage of still alive nodes vs. number of transmitted packets comparing q-switch, Maximizing, and GoS.

packets to the sink node, extending the network lifetime. Moreover, GoS is not tied to fixed network deployments that do not always suit all the sensing applications well. In addition, the *q-switch* and Maximizing models require a constant bit rate to guarantee an energy balance among the coronas. Finally, the Maximizing technique requires that sensor nodes have a different battery capacity, which is not always possible. Contrary to these algorithms, GoS uses a straightforward technique that can be easily implemented in sensors with limited resources.

VI. CONCLUSION

In contrast to related works in which most authors use complicated methods to extend the network lifetime, such as clustering and corona-based algorithms, GoS takes advantage of one of the most interesting cellular automatons known as the Game of Life. The conjunction of this simple model, a variation of the A-star algorithm, and the new set of rules enhance the network lifetime compared to the most recent works such as [4], [5], [17], [20], and [40], resulting in a large number of transmitted packets preserving the simplicity of cellular automatons models. Moreover, GoS can be easily implemented on limited-resource sensors since the set of proposed transition rules only requires simple programming techniques. In addition, the proposed algorithm considers that parameters such as the remaining energy, the number of active neighbors, and a sleep schedule guarantee energy balance while minimizing the number of hops to reach the sink node through the A-star algorithm.

Furthermore, simulation results demonstrate that the proposed model accurately balances energy consumption, extending the network lifetime compared to the most recent works. Moreover, we envision the proposed method as a suitable algorithm that can be combined with path planning algorithms. Finally, for future research, the proposed technique can use mobile sinks on a cellular automaton model to maximize energy consumption, especially for sparse wireless sensor networks.

CONFLICTS OF INTEREST

The authors declare that there is no conflict of interest regarding the publication of this paper.

REFERENCES

[1] O. Arana, F. Garcia, and J. Gomez, "Analysis of the effectiveness of transmission power control as a location privacy technique," *Comput. Netw.*, vol. 163, Nov. 2019, Art. no. 106880. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128618311034
[2] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad Hoc Netw.*, vol. 7, no. 3, pp. 537–568, May 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870508000954
[3] H.-W. Ferng, M. Hadiputro, and A. Kurniawan, "Design of novel node distribution strategies in corona-based wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 10, no. 9, pp. 1297–1311, Sep. 2011.
[4] X. Wu, G. Chen, and S. K. Das, "Avoiding energy holes in wireless sensor networks with nonuniform node distribution," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 5, pp. 710–720, May 2008.
[5] H. Asadollahi, S. Zandi, and H. Ashariоun, "Maximizing network lifetime in many-to-one wireless sensor networks (WSNs)," *Wireless Pers. Commun.*, vol. 123, no. 4, pp. 2971–2983, Apr. 2022, doi: 10.1007/s11277-021-09271-9.
[6] C. Banerjee and S. Saxena, "Energy conservation in wireless sensor network using block cellular automata," in *Proc. Int. Conf. Comput. Commun. Informat.*, Jan. 2013, pp. 1–6.
[7] H. Byun and J. Yu, "Cellular-automaton-based node scheduling control for wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 63, no. 8, pp. 3892–3899, Oct. 2014.
[8] C. P. Subha and S. Malarkkan, "Optimization of energy efficient cellular learning automata algorithm for heterogeneous wireless sensor networks," in *Proc. 10th Int. Conf. Intell. Syst. Control (ISCO)*, Jan. 2016, pp. 1–6.
[9] S. Choudhury, "Cellular automata and wireless sensor networks," in *Emergent Computation*. Cham, Switzerland: Springer, 2017, pp. 321–335.
[10] S. Hao, H. Zhang, and M. Song, "A stable and energy-efficient routing algorithm based on learning automata theory for MANET," *J. Commun. Inf. Netw.*, vol. 3, no. 2, pp. 43–57, 2018.
[11] P. S. Khot and U. L. Naik, "Cellular automata-based optimised routing for secure data transmission in wireless sensor networks," *J. Experim. Theor. Artif. Intell.*, vol. 34, no. 3, pp. 431–449, May 2022.
[12] J. Lee, S. Adachi, F. Peper, and K. Morita, "Asynchronous game of life," *Phys. D, Nonlinear Phenomena*, vol. 194, nos. 3–4, pp. 369–384, Jul. 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167278904001198
[13] T. Fujita, K. Nakano, and Y. Ito, "Fast simulation of Conway's game of life using bitwise parallel bulk computation on a GPU," *Int. J. Found. Comput. Sci.*, vol. 27, no. 8, pp. 981–1003, Dec. 2016.
[14] I. Gupta, D. Riordan, and S. Sampalli, "Cluster-head election using fuzzy logic for wireless sensor networks," in *Proc. 3rd Annu. Commun. Netw. Services Res. Conf. (CNSR)*, May 2005, pp. 255–260.
[15] Y. Yuan, C. Li, Y. Yang, X. Zhang, and L. Li, "CAF: Cluster algorithm and A-star with fuzzy approach for lifetime enhancement in wireless sensor networks," *Abstract Appl. Anal.*, vol. 2014, pp. 1–17, Jan. 2014.
[16] G. Smaragdakis, I. Matta, and A. Bestavros. (2004). *SEP: A Stable Election Protocol for Clustered Heterogeneous Wireless Sensor Networks*. [Online]. Available: https://open.bu.edu/handle/2144/1548
[17] A. Ghaffari, "An energy efficient routing protocol for wireless sensor networks using A-star algorithm," *J. Appl. Res. Technol.*, vol. 12, no. 4, pp. 815–822, 2014.
[18] S. Soijoyo and R. Wardoyo, "Wireless sensor network energy efficiency with fuzzy improved heuristic A-Star method," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 4, pp. 1–7, 2017.
[19] I. S. AlShawi, L. Yan, W. Pan, and B. Luo, "Lifetime enhancement in wireless sensor networks using fuzzy approach and A-star algorithm," in *Proc. IET Conf. Wireless Sensor Syst. (WSS)*, 2012, pp. 1–6.
[20] A. A. Alkadhmawee and S. Lu, "Prolonging the network lifetime based on LPA-star algorithm and fuzzy logic in wireless sensor network," in *Proc. 12th World Congr. Intell. Control Autom. (WCICA)*, Jun. 2016, pp. 1448–1453.
[21] E. Amiri, H. Keshavarz, M. Alizadeh, M. Zamani, and T. Khodadadi, "Energy efficient routing in wireless sensor networks based on fuzzy ant colony optimization," *Int. J. Distrib. Sensor Netw.*, vol. 10, no. 7, Jul. 2014, Art. no. 768936.
[22] A. J. Obaid, "Wireless sensor network (wsn) routing optimization via the implementation of fuzzy ant colony (faco) algorithm: Towards enhanced energy conservation," in *Next Generation of Internet of Things*, R. Kumar, B. K. Mishra, and P. K. Pattnaik, Eds. Singapore: Springer, 2021, pp. 413–424.

[23] S. Tanwar, S. Tyagi, N. Kumar, and M. S. Obaidat, "LA-MHR: Learning automata based multilevel heterogeneous routing for opportunistic shared spectrum access to enhance lifetime of WSN," *IEEE Syst. J.*, vol. 13, no. 1, pp. 313–323, Mar. 2019.
[24] A. H. F. Navid, "SELARP: Scalable and energy-aware learning automata-based routing protocols for wireless sensor networks," in *Proc. 4th Int. Conf. Sensor Technol. Appl.*, Jul. 2010, pp. 570–576.
[25] M. Kamarei, A. Patooghy, Z. Shahsavari, and M. J. Salehi, "Lifetime expansion in WSNs using mobile data collector: A learning automata approach," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 32, no. 1, pp. 65–72, Jan. 2020.
[26] T. Liu, T. Gu, N. Jin, and Y. Zhu, "A mixed transmission strategy to achieve energy balancing in wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 4, pp. 2111–2122, Apr. 2017.
[27] G. Chen, C. Li, M. Ye, and J. Wu, "An unequal cluster-based routing protocol in wireless sensor networks," *Wireless Netw.*, vol. 15, no. 2, pp. 193–207, Feb. 2009, doi: 10.1007/s11276-007-0035-8.
[28] G. M. B. Oliveira, R. G. O. Silva, G. B. S. Ferreira, M. S. Couceiro, L. R. do Amaral, P. A. Vargas, and L. G. A. Martins, "A cellular automata-based path-planning for a cooperative and decentralized team of robots," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2019, pp. 739–746.
[29] L. G. A. Martins, R. D. P. Cândido, M. C. Escarpinati, P. A. Vargas, and G. M. B. de Oliveira, "An improved robot path planning model using cellular automata," in *Towards Autonomous Robotic Systems*, M. Giuliani, T. Assaf, and M. E. Giannaccini, Eds. Cham, Switzerland: Springer, 2018, pp. 183–194.
[30] S. C. S. Nametala, L. G. A. Martins, and G. M. B. Oliveira, "A new distance diffusion algorithm for a path-planning model based on cellular automata," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2020, pp. 1–8.
[31] G. B. S. Ferreira, P. A. Vargas, and G. M. B. Oliveira, "An improved cellular automata-based model for robot path-planning," in *Autonomous Robotics Systems*, M. Mistry, A. Leonardis, M. Witkowski, and C. Melhuish, Eds. Cham, Switzerland: Springer, 2014, pp. 25–36.
[32] J. Tariq and A. Kumaravel, "Construction of cellular automata over hexagonal and triangular tessellations for path planning of multi-robots," in *Proc. IEEE Int. Conf. Comput. Intell. Comput. Res. (ICCIC)*, Dec. 2016, pp. 1–6.
[33] E. M. Izhikevich, J. H. Conway, and A. Seth, "Game of life," *Scholarpedia*, vol. 10, no. 6, p. 1816, 2015.
[34] R. B. Elwin, J. H. Conway, and R. K. Guy, *Winning Ways for Your Mathematical Plays*. Boca Raton, FL, USA: CRC Press, 2018, doi: 10.1201/9780429487330.
[35] M. G. Kechaidou and G. C. Sirakoulis, "Game of life variations for image scrambling," *J. Comput. Sci.*, vol. 21, pp. 432–447, Jul. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877750316301442
[36] S. M. Lucas, A. Dockhorn, V. Volz, C. Bamford, R. D. Gaina, I. Bravi, D. Perez-Liebana, S. Mostaghim, and R. Kruse, "A local approach to forward model learning: Results on the game of life game," in *Proc. IEEE Conf. Games (CoG)*, Aug. 2019, pp. 1–8.
[37] X. Wang and C. Jin, "Image encryption using game of life permutation and PWLCM chaotic system," *Opt. Commun.*, vol. 285, no. 4, pp. 412–417, Feb. 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0030401811010893
[38] A. Adamatzky, Ed., *Game Life Cellular Automata*, 1st ed. London, U.K.: Springer, 2010.
[39] T. Shankar, G. Eappen, and S. Rajalakshmi, "Optimized routing algorithm for wireless sensor networks," in *Game of Life Cellular Automata*. Singapore: Springer, 2021, pp. 83–96.
[40] I. S. AlShawi, L. Yan, W. Pan, and B. Luo, "Lifetime enhancement in wireless sensor networks using fuzzy approach and A-star algorithm," *IEEE Sensors J.*, vol. 12, no. 10, pp. 3010–3018, Oct. 2012.

JOEL REYES was born in Mexico, in 1997. He received the B.S. degree (Hons.) in telecommunications engineering from the National Autonomous University of Mexico, in 2020. He is currently pursuing the M.S. degree in computer science and engineering with the Department of Telecommunications, UNAM, Mexico. He joined the Signal Integrity Department, Intel, in 2022, where he has been involved in software optimization. His research interests include cellular automata models, sensor networks, and machine learning.

FRANCISCO GARCÍA received the B.S. degree in electrical engineering from the National Autonomous University of Mexico (UNAM), in 2005, and the M.S. and Ph.D. degrees (Hons.) in computer science from the IIMAS, UNAM, in 2007 and 2015, respectively. He worked at The European Organization for Nuclear Research (CERN) on the ALICE project. He is currently a full-time Professor at the Department of Telecommunications Engineering, UNAM. His research interests include sensor networks, SDN security, and localization techniques. He has been a member of the SNI, since 2017.

MARÍA ELENA LÁRRAGA received the B.S. degree in computer science from the Benemérita Universidad Autónoma de Puebla, Mexico, and the master's degree in computer science and engineering and the Ph.D. degree in system engineering (transport) from the Universidad Nacional Autónoma de México. She is currently working as a Researcher with the Universidad Nacional Autónoma de México. Her research interests include modeling and simulation of complex systems, such as traffic flow analysis, epidemics, security, and malware propagation as well as cellular automata and agent-based models. She served as a TPC member and a reviewer for many leading international conferences and journals.

JAVIER GÓMEZ received the B.S. degree (Hons.) in electrical engineering from the National Autonomous University of Mexico (UNAM), in 1993, and the M.S. and Ph.D. degrees in electrical engineering from the COMET Group, Columbia University, in 1996 and 2002, respectively. During his Ph.D. studies at Columbia University, he collaborated and worked on several occasions at the IBM T. J. Watson Research Center, Hawthorne, NY, USA. He is currently a full-time Professor at the Department of Telecommunications Engineering, School of Engineering, UNAM. His research interests include routing, QoS, and MAC design for wireless ad hoc, sensor, and mesh networks. He has been a member of the SNI, since 2004.

LUIS OROZCO-BARBOSA (Member, IEEE) received the Diplôme d'Etudes Approfondies degree in computer science from the École Nationale Supérieure d'Informatique et de Mathématiques Appliquées, France, in 1984, and the Doctorat d'Université degree in computer science from the Université Pierre et Marie Curie, France, in 1987. From 1991 to 2002, he was a Faculty Member with the School of Information Technology and Engineering, University of Ottawa, Canada. Since 2002, he has been a Professor with the Department of Computer Engineering, Universidad de Castilla-La Mancha, Spain, and the Director of the Albacete Research Institute of Informatics. He has conducted numerous research and innovation projects with the private sector, contributed to ITU standards. His current research interests include the IoT technologies, B5G networks, modeling, and performance evaluation. He is a member of the IEEE Communications Society and various COST actions in wireless technologies. He served as a Technical Advisor for the Canadian International Development Agency and the Spanish International Cooperation Council.