A High-robust Sensor Activity Control Algorithm for Wireless Sensor Networks

Rong-Guei Tsai
Putian University
Pei-Hsuan Tsai (peihsuan.tsai@gmail.com)
National Cheng Kung University https://orcid.org/0000-0001-6227-7652

Research Article

Keywords: quality of service, Gur game, wireless sensor networks, internet of things, power saving

Posted Date: December 13th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-1097637/v1

Version of Record: A version of this preprint was published at Sensors on March 4th, 2022. See the published version at https://doi.org/10.3390/s22052020.

A High-robust Sensor Activity Control Algorithm for Wireless Sensor Networks

Rong-Guei Tsai[1] and Pei-Hsuan Tsai[2*]

*Correspondence:
phtsai@mail.ncku.edu.tw
1 School of Computer Science and Technology, National ChengKung University, Taiwan. Full list of author Information is available at the end of the article.

Abstract

In wireless sensor networks, it is important to use the best number of sensors to optimize the network and consider the key design and cost. Owing to the limited power of sensors, how controlling the state of the sensor through an automatic control algorithm and power-saving and efficient distribution of work have become important issues. However, sensor nodes are usually deployed in dangerous or inaccessible locations. Therefore, it is difficult and impractical to supply power to sensors through humans. In this study, we propose a high-reliability control algorithm with fast convergence and strong self-organization ability, called sensor activity control algorithm (SACA), which can efficiently control the number of sensors in the active state and extend their use time. SACA considers the relationship between the total number of inactive sensors and the target value and determines the state of the sensor in the next round. The data transmission technology of random access is used between the sensor and the base station; therefore, the sensor in the sleep state does not need to receive the feedback packet from the base station. The sensor can achieve true dormancy and power-saving effects. The experimental results show that SACA has fast convergence, strong self-organization capabilities, and power-saving advantages.

Keywords: quality of service; Gur game; wireless sensor networks; internet of things; power saving

1. Introduction

1.1 Background

Wireless sensor networks (WSNs) are one of the critical technologies of the Internet of Things and are mainly composed of base stations and hundreds or thousands of sensors. Because the power of sensors is limited, and sensors are usually deployed in dangerous or inaccessible locations. It is difficult and impractical to supply power

to sensors in areas that are not easy for humans to reach. Therefore, controlling the state of the sensor through an automatic control algorithm and maximizing the function of the sensor, power saving, and efficient distribution of the mission have become important issues. Thus, efficiently controlling the number of sensors in the active state improves their working strategy and efficiency and extends their use time [1-4]. Moreover, the number of sensors in the active state should be controlled according to the application requirements. For example, the number of sensors required to monitor the temperature of a reservoir or air pollution in the environment are small. In contrast, for applications that require higher accuracy, such as military and disaster area monitoring, the number of sensors can be increased to thousands.

However, it is difficult to control the number of active sensors to a certain number. Owing to the large number of sensors and the wide distribution area, it is not feasible to obtain information from all sensors for control. In addition, a large number of active sensors waste power and bandwidth, whereas few active sensors result in insufficient environmental information. In an actual network environment, owing to battery failure, sensor damage, and the addition of sensors, the total number of sensors changes frequently.

To address these issues, in [5], the authors first applied the Gur Game algorithm to control the number of active sensors and defined the quality of service as the number of active sensors. In the Gur game, each active sensor sends a packet to the base station. When the base station receives packets from the active sensors, it counts the total number of packets and calculates the probability of the sensor changing the state in the next round through the reward function. After several operations, the total number of active sensors gradually reaches the target value. In [6], the authors proposed the ACK algorithm, in which the base station individually sends feedback packets to each active sensor. Sleep sensors do not need to receive feedback packets, and the ACK algorithm uses a random-access mechanism to increase the lifetime of the entire network [6-10].

In this study, we propose a high-reliability control algorithm with fast convergence and strong self-organization ability, called the sensor activity control algorithm (SACA). In SACA, there are two sub-processes: the adjustment procedure for the weight value of sensors and the transformation procedure of the state of

sensors. SACA considers the relationship between the total number of active sensors and the target value and determines the state (ON or OFF state) of the sensors in the next round. Random access transmission technology is used between the sensor and the base station such that the sleep sensors do not need to receive the feedback packet from the base station, resulting in power savings.

1.2 Contributions

The three contributions of this research are as follows:

(1) This paper proposes a sensor control algorithm with a fast convergence. According to the application requirements, an algorithm that can provide fast convergence can avoid the waste of sensor power resources. SACA also uses random access technology, such that the sleep sensors do not need to receive the feedback packet from the base station and can truly achieve the purpose of dormancy and power saving. In the Gur game algorithm, all sensors must receive feedback packets from the base station to switch their states.

(2) The SACA has a high scalability algorithm and can be applied in a large-scale network environment. The sensor network is often composed of hundreds or thousands of sensor nodes. In Gur Game, when the ratio of the target value to the total number of sensors is too large or too small, the number of active sensors cannot reach the target value.

(3) SACA has a strong self-organization ability algorithm and can be applied in severe network environments, such as a network environment with high death (high damage rate). Because sensors are usually deployed in dangerous areas, it is not possible to repair or add new sensors. Therefore, an algorithm with strong self-organization capabilities can enable sensors to operate in a severe network environment in real time, self-adjust, or maintain a specific target value.

The remainder of this paper is organized as follows. Section 2 describes the Gur game and ACK approaches, as well as the QoS control used to solve different problems. Section 3 describes the preliminaries and assumptions of SACA. Section 4 describes the operational details of SACA. Section 5 presents the experimental analysis of the SACA, ACK, and Gur games. The convergence time, success rate, and self-organization capability of the sensor nodes of the algorithms were compared individually. The final section concludes the paper.

2. Background

2.1 Gur Game Algorithm

In 2003, the Gur game algorithm was first applied to control the number of active sensors [5]. In the Gur game algorithm, it is assumed that the base station and sensor communicate in a signal hop, and they are time synchronized. Each sensor had a state machine. Before the Gur Game algorithm starts to operate, the sensor selects a state from the state machine as its initial state, and states 2 and 1 represent the active state, and -2 and -1 are sleep states. The sensor must receive a data packet from the base station, regardless of whether it is in an active or sleep state.

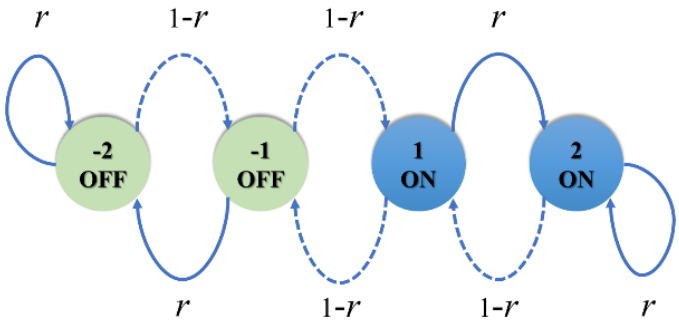


Fig. 1 State machine used by the sensor in the Gur Game algorithm.

In each round, each active sensor sends a packet to the base station. The base station counts the total number of packets received and then calculates the probability value $r(t)$ of the state transition through the reward function, as shown in (1), where $0 \leq r(t) \leq 1$. Q_1 is the total number of packets received at time t, and Q_0 is the target value. After the base station calculates $r(t)$, it broadcasts the feedback packets to all the sensor nodes. After receiving the feedback packets, the sensor randomly generates a random value R of 0–1. When the random value R is less than $r(t)$, the sensors move to the 2 or -2 state of the state machine. Otherwise, the sensors move to the 1 or -1 state in the state machine. The sensor then determines the state of the next round based on the cur-rent position in the state machine. After several rounds of operation of the algorithm, the number of active sensors reaches (converges to) the target value.

$$r(t)=0.2+0.8\exp(-0.002(Q_1-Q_0)^2) \quad (1)$$

Fig. 2 shows the results corresponding to the reward function of the Gur game approach, where the total number of sensors m is 100 and the target value Q_0 is 35. After approximately 1200 epochs, the system achieved the target value. When

the value of $r(t)$ is large, the sensor has a high probability of maintaining its original state. For example, the target value was set to 35. After several runs, the number of active nodes achieved the target values, the calculated probability value $r(t)$ approached 1, and the state of the sensor nodes no longer changed.

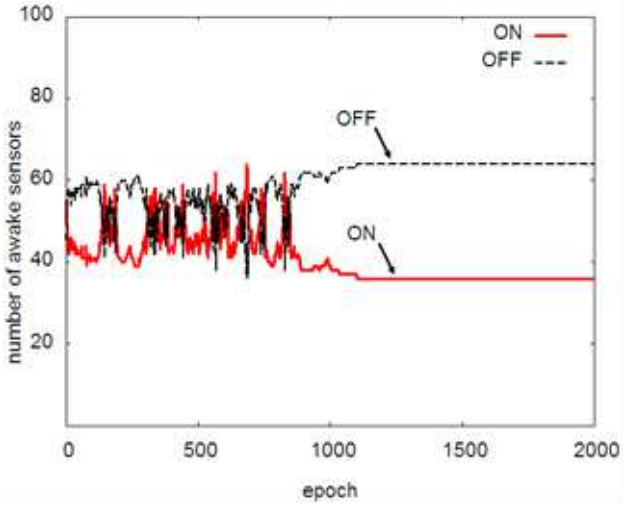


Fig. 2 Execution results of the Gur game approach when the target value Q_0 is set as 35.

2.2 ACK Algorithm

Another control method is the ACK algorithm [6]. For the Gur Game algorithm, owing to the limited power of the sensors, the state of the sensors does not change when the number of active sensors reaches the target value, which results in low power utilization efficiency. In the ACK algorithm, using random-access technology, the base station individually re-turns feedback packets to the active sensors. Therefore, sleep sensors do not need to receive feedback packets from the base station.

The ACK algorithm also does not use a reward function to control the state of the sensor. Each sensor had a state machine, as shown in Fig. 3. In the ACK algorithm, the state machine has three states, and each state contains the probability of a sensor being awakened (P_1, P_2, and P_3). Sensors in the P_3 state have a higher probability of being awakened than sensors in the P_2 state, which have a higher probability than sensors in the P_1 state ($P_3 > P_2 > P_1$). The probabilities were 1.00, 0.10, and 0.05, respectively.

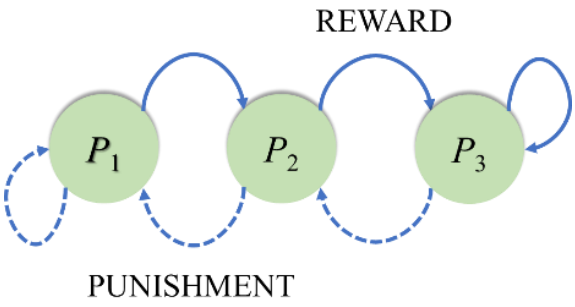


Fig. 3 Finite-state machine in the ACK algorithm

When the ACK algorithm starts to operate, each active sensor randomly selects a state as the initial state. Each sensor selects an active or sleep state based on the probability P_i of its current state. The base station receives packets from active sensors, counts the number of packets received, and sends feedback packets to the active sensors individually. Q_1 refers to the total number of packets received, and Q_0 is the target value. The feedback packet contains the relationship between Q_0 and Q_1.

The base station, according to the relationship between the number of active sensors and the target value, indicates the sensors to change their state. When the number of packets (total number of active sensors) is higher than the target value, the feedback packet indicates that the sensors should move to the left of the state machine. Conversely, if the number of packets is less than or equal to the target value, the feedback packet indicates that the sensor should move to the right side of the state machine. When all the active sensors received the feedback packet, a round was completed. In the next round, the sensor decides between the ON and OFF states according to probability P_i. After several runs, the number of active sensors gradually reaches the target value. Fig. 4 shows the experimental results, where the total number of sensors M was set to 100, and the target value was set to 35. The experimental results show that the number of active sensors oscillates in the range of 34–42.

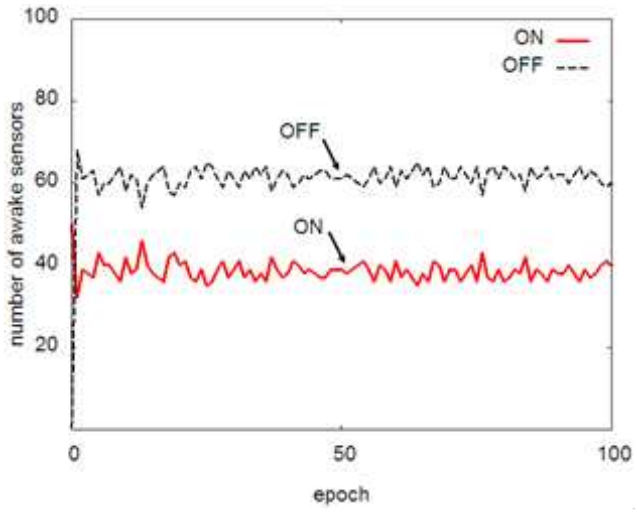


Fig. 4 Execution result of the ACK approach when the target value is set as 35.

2.3 Related Works

Many studies have been conducted on the different aspects of QoS of sensor networks, such as power-saving [10-15], routing [16-19], coverage [22-25], and throughput of networks [26]. The main goal of this study was to autonomously control the number of active sensors. Considering that the sensor network may be in a harsh environment, a feasible approach must be able to reach the goal quickly, have strong robustness, and power-saving properties to cope with the harsh network environment.

Distributed control using finite state automata to control the number of active sensors can be divided into: (1) Gur game-based and (2) random access based. In the Gur game, when the number of active sensors (QoS) reaches the target value, the sensor does not change its state, resulting in uneven power usage of the sensors. In [5], the network lifetime is defined as when the power of the first sensor node is exhausted, and the lifetime ends. In [6], the system begins to execute until the number of active sensors cannot maintain the target value, and the network lifetime ends. In [10-15], the authors aim to solve the problem of sensor power imbalance and maximize the lifetime of the network system regardless of the definition. To solve the problem of Gur game in terms of power imbalance, the mechanism of using sensors to exchange states at a predetermined time can improve the network lifetime [11]. However, the sensor undergoes state exchange without considering the remaining power of the

sensors and does not overcome the power imbalance problems. Therefore, in [12], the authors improved the state transition strategy in the finite state machine and modified the reward function for the relationship between QoS and the target value, aiming to speed up the convergence speed and make the power of the sensor even. In [13], the authors proposed a new adaptive method for QoS and energy management in IEEE 802.15.4 networks, called skip game, which aims to maintain a trade-off between QoS and extend the lifetime of networks.

In [14], the authors proposed a QoS control algorithm called the Gureen game. A Gureen game divides the sensor state into active, standby, and sleep states. The standby state is the same as the Gur game sleep state, whereas the sleep state turns off all the functions of the sensor, depending on whether the QoS is too many or too few. Adjust the reward probability of the situation. Each sensor uses a timer to change its state to achieve power-saving effects. Similar to the Gureen game, in [15], the sensor state is divided into *T1*, *T2*, *R*, and *S* states, and the sensor in the *S* state will turn off all functions, thereby allowing the sensor to achieve true dormancy, and the sensor in the *S* state will change to the *T1* state at the desired time to achieve the purpose in terms of power balance.

Fast convergence can prevent the sensor from consuming unnecessary power resources and bandwidth. [20] proposed a fast convergence control method, called QC^2, which is mainly based on the relationship between the QoS and the target value, and using the virtual target, shortens the convergence time. However, this method faces an environment in which the sensors continue to die. Owing to continuous changes in QoS, the aim of power saving cannot be achieved.

In the multi-objective QoS control algorithm, the authors combined the two control objectives of QoS and coverage and proposed a compound control method that can control the coverage rate in addition to QoS [22]. In [23-25], combining the two control objectives of QoS and node density, the control strategies for rectangular, circular, and various elliptical regions are given, and a dynamic adjustment mechanism based on the Gur game algorithm is designed. In general, the convergence time of algorithms based on the Gur game is slow and cannot be adapted to a large-scale network environment; when the QoS continues to change, owing to the slow convergence time, it will not be able to achieve the desired target value.

3. Preliminaries

Sensor power resources are limited, making energy supply in harsh environments difficult. In the Gur game algorithm, the sink broadcasts feedback packets to all the sensors. Power resources are consumed when the sensors receive packets. Conversely, some studies show that the energy consumption of the transmit and receive data are almost the same 5. In the ACK algorithm, the sink transmits an acknowledgment message to the active sensor. Sleep sensors do not require feedback packets. Sleep sensors are in a low-energy consumption state to avoid unnecessary waste of power resources. In this study, we proposed a fast convergence, robust self-organization, and highly reliable control method of QoS called SACA.

Similar to the ACK approach, in the SACA the sink individually transmits QoS packets to each active sensor Sleep sensors are in a low energy consumption state to avoid unnecessary waste of power resources. Meanwhile, the sink uses variations in the number of active sensors. The sensors are rewarded when they benefit the system results and are punished when they do not. Thus, we expect to determine the number of active sensors required to achieve the target value. If the sensor is rewarded, it has a high probability of being in a sleep state. Otherwise, the punished sensors will be in the next round and will have a high probability of being in an active state.

3.1. Sensor activity control algorithm

This study proposes the SACA, which is a sensor control algorithm with high robustness, fast convergence, and power saving. The sensor and base stations use random-access technology to send and receive packets. The base station calculates the difference d between the total number of packets received at the current time (round) and the target value, and adds the difference d in the feedback packet, which is sent to the sensor. The sensor uses a weight value to determine whether it should switch its state. SACA mainly has two sub-processors to control the number of active sensors, namely, the sensor adjustment procedure for the weight value of sensors and the transformation procedure of the state of sensors. In the state transition process, the active sensor determines whether it should continue in the active state or switch to the sleep state in the next round based on the difference d recorded in the feedback packet. Sleep sensors do not need to receive feedback packets from the base station to

achieve true dormancy. The sleep sensors gradually increase their weight value in each round to increase the probability of changing to the active state. In the adjustment procedure for the weight value of the sensors, the sensor decides whether to change the state or maintain the original state in the next round according to its weight value W_i.

Assume that m sensors and a base station are randomly deployed in the monitoring area. Before the system starts operating, the sensor randomly selects a state (e.g., active or sleep state). Each sensor contained a weight value W_i with an initial value of 0. The weight value W_i affects the probability that the sensor changes its state in the next round. Each packet is transmitted from the sensor to the base station in one hop, and vice versa.

3.2 Operation Process of SACA

The following four steps comprise the entire operation process of SACA:

Step 1: Each active sensor sends a packet to the base station. Sleep sensors do not need to send packets to the base station.

Step 2: The base station starts to count the total number of packets from the active sensors, determine whether the total number of active sensors Q_1 is close to the target value Q_0, and calculates the difference value d between Q_1 and Q_0. The difference value d is added to the feedback packet and sent to the active sensor.

Step 3: The sensor executes the adjustment procedure for the weight values of the sensors. The active sensor determines whether to change its state (for example, maintain the original state or change from the ON state to the OFF state) according to the difference value d in the feedback packet. If the number of active sensors is too small (i.e., Q_1-Q_0 < 0), the sensors should remain in their original state. If the number of active sensors is too large in the current round (i.e., Q_1-Q_0 > 0), the sensor should change to the sleep state. If the difference value d is equal to 0 (i.e., Q_1-Q_0 = 0), the sensors are maintained in the original state. Sleep sensors do not need to receive feedback packets from the base station. They increase their weight value W_i in each round, as their main purpose is to turn to the active state to replace the active sensors.

Step 4: After the sensors finish the adjustment procedure for their weight value, they execute the change-of-state procedure. The sensors randomly generate a

probability value r and compare it with their own weight value W_i. When $W_i > r$, the sensor switches to the operating state; otherwise, it changes to the sleep state.

The four steps mentioned above, shown in Fig. 5, represent one round. After the system executes several rounds, the total number of active sensors converges to the target value.

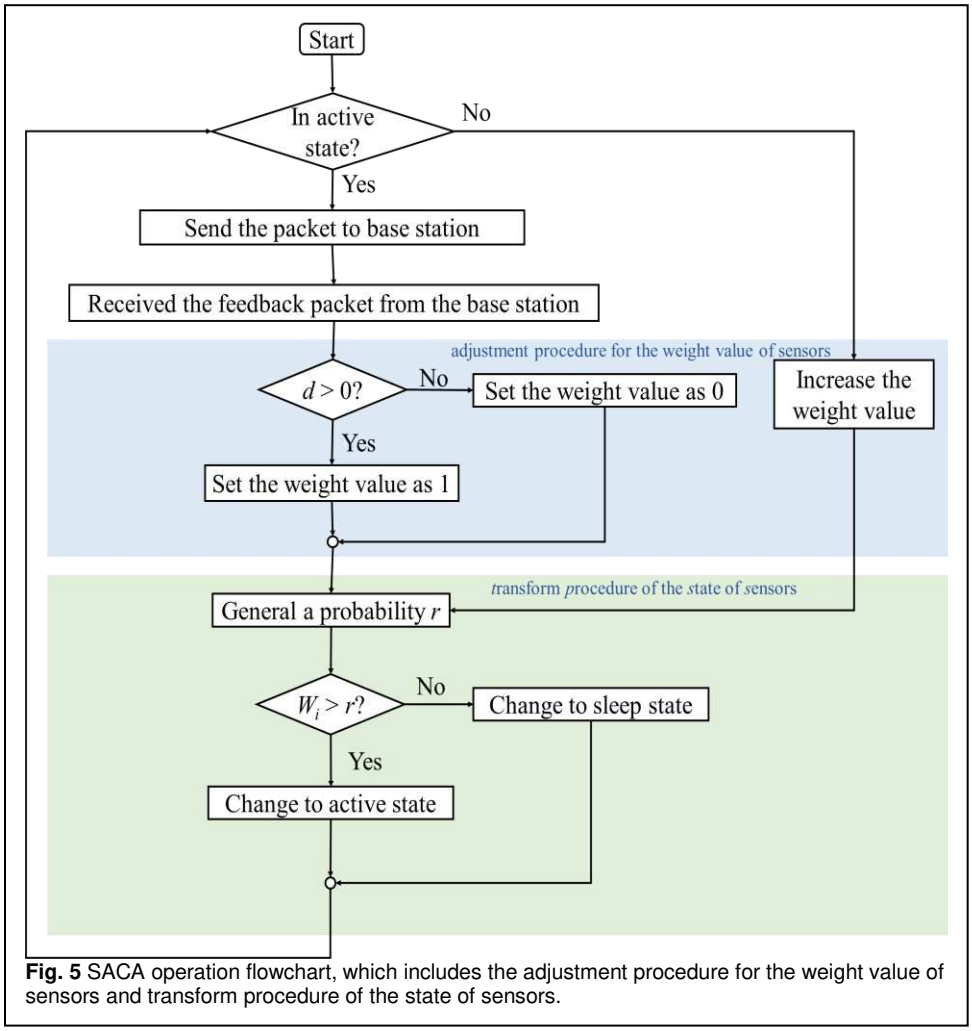


Fig. 5 SACA operation flowchart, which includes the adjustment procedure for the weight value of sensors and transform procedure of the state of sensors.

4. Methods

4.1 Adjustment Procedure for the Weight Value of Sensors

As previously mentioned, a sensor with a higher weight value, W_i, will have a higher probability of maintaining the active state. Therefore, SACA uses the relationship between the total number of active sensors and the target value to adjust the weight of sensor W_i. For the active sensors, the difference value d is used to determine whether the weight value W_i should be increased or decreased (reward or punishment), where d is the difference between the target value and the total number of active sensors. The total number of active sensors is counted

using the base station, which records the difference value d in the feedback packet and sends it to the sensor according to the relationship between the total number of packets currently counted and the target value.

Only the active sensors receive feedback packets from the base station in the SACA. When the sensor is in the active state, the following three situations need to be considered.

(1) When $d > 0$, there are too many sensors in the active state. Because the sensor node is currently in an active state, the sensor should change to a sleep state in the next round.

(2) When $d < 0$, the total number of active sensors is currently too small. If the sensor is currently in an active state, it should be maintained in an active state in the next round.

(3) When $d = 0$, the total number of active sensors reached the target value. Therefore, the state of the sensor does not need to be changed. These situations are shown in Fig. 6.

For the sensor in the sleep state, the weight value W_i automatically increases every round. The main reason for this is that the sensor in the sleep state can replace the active sensors such that the active sensors can rest. The power loss between the sensors was then averaged. The smaller the weight value W_i, the higher the probability of the sensor remaining in the sleep state or transition from the active state to the sleep state. Therefore, increasing the weight of the sensor increases the probability that the sensor is in an active state. The weight value W_i is updated as shown in (2). The activation parameter of the weight value W_i is σ, which affects the convergence time of the system. If the activation parameter σ is small, the weight value of the sensor needs a longer time to adjust, and the sensor needs to spend more time before it can be converted into an active state. In contrast, if the activation parameter σ is large, the probability of the sensor turning to the active state increases. The sensor in the sleep state can usually be used to supplement other sensors in the active state, for example, when a sensor in the active state is damaged or dead. Therefore, in a network environment with a high death rate, setting the activation parameter σ is very important. In SACA, we set the activation parameter σ to 0.0001.

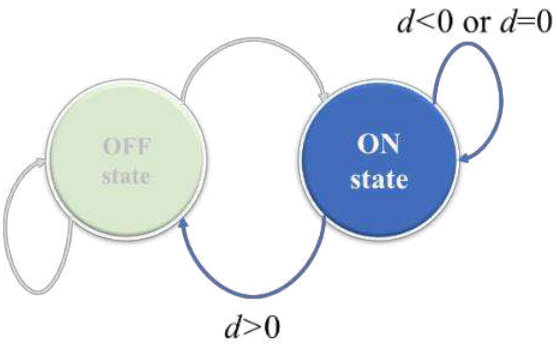


Fig. 6 The active sensor performs a state transition according to the difference value d in the feedback packet.

4.2 Transform Procedure of the State of Sensors

Each sensor was set with a weight value of W_i, which was between 0 and 1. Before the system started to operate, the initial weight value of each sensor was set to 0. The weight value W_i affects the state of the sensor in the next round. In the transformation procedure of the state of the sensors, the sensor generates a probability value r between 0 and 1. The state transition rules are described as follows:

(1) When r is higher than W_i, the sensor remains in the sleep state or transitions from the active state to the sleep state.

(2) When r is lower than W_i, the sensor remains in the active state or transitions from the sleep state to the active state. In other words, a sensor node with a higher weight value W_i will have a higher probability of being in the active state.

We implemented the SACA and set the activation parameters to 0.0001. Fig. 7 shows the execution result when the total number of sensors is 100, and the target value is set to 35.

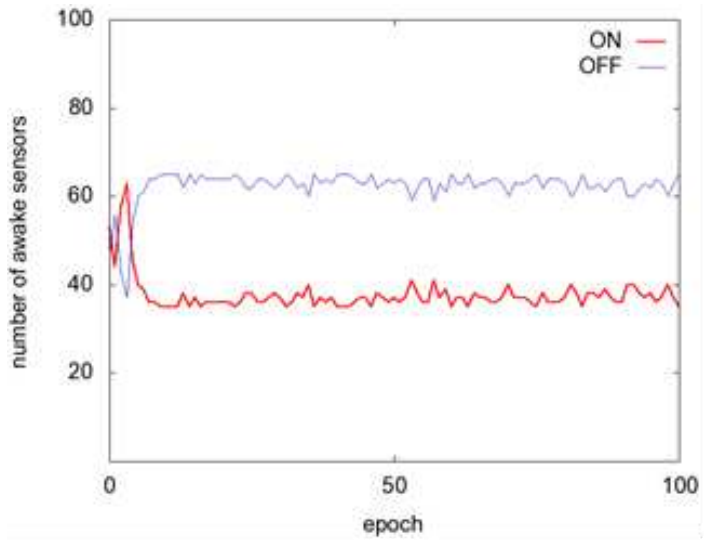


Fig. 7 Execution result of the SACA when the target value is set as 35.

Next, we summarize the approaches and features of the ACK and SACA. SACA and ACK are based on QoS packets from the active sensors. Sleep sensors do not have to receive QoS packets. Sleep sensors are in a low energy consumption state to avoid unnecessary waste of power resources. The ACK and SACA differ in that the former uses a state machine with a fixed probability value as the probability of the sensor state. ACK uses variations in the number of active sensors and target value. The position of the sensor was converted into a finite-state machine to adjust the probability of the active state. In the SACA, every sensor was set to a weight value. SACA uses the variation d in the number of active sensors and the target value to adjust the probabilities of the active and sleep modes.

5. Evaluations and Results

We designed three sets of experiments to verify the performance of SACA: convergence time, success rate, large-scale network environment, and power loss. We observed that the active sensor quickly converged to the target value, avoiding the waste of power resources. In Section 5.2, we compare the convergence capabilities of the SACA, Gur Game, and ACK algorithms. We analyzed the performance of the three algorithms in terms of convergence time and success rate to reach the target value. In Section 5.3, we applied the three algorithms to a severe network environment to verify self-organization ability. In Section 5.4, the power-saving capabilities of the three algorithms are compared. We also provide suggestions for improving SACA performance.

5.1. Performance Measurement

Equation (3) shows the calculation algorithm for average convergence time. In E trials, the sum of the convergence time of successful convergence was averaged, where c_j and s are the convergence time and number of successes with the target value set to j, respectively. Equation (4) is the calculation algorithm for the ratio of successful convergence S_i, that is, the percentage of successful convergence times in E trials. In the SACA, the activation parameters were set to 0.001 and 0.0001, respectively. Table 1 lists the experimental parameter settings.

$$T = \frac{\sum_{j=0}^{m} c_j}{Suc} \qquad (3)$$

Table 1 Experimental parameter settings

Items	**Parameter**
Total number of sensors, m	10^2, 10^3-10^4
Number of the sink	1
Target value Q_0	0-100 0.3, 0.7(Q_0/m)
Adjustment parameter, $\Delta\tau$	1
Evoke parameter, σ	0.001, 0.0001
Number of experiments	100
execute rounds for each set of experiment	3000

5.2 Convergence Time and Successful Rate Analysis

Fig. 8 shows the execution result of the Gur game; the target value was increased from 0 to 100. It can be observed from the experimental results that convergence to the target value occurred only when the target value was between 25 and 75. Therefore, when the target value is set too large or too small, the number of active sensors cannot successfully converge to the target value. In other words, the Gur game has restrictions on its use.

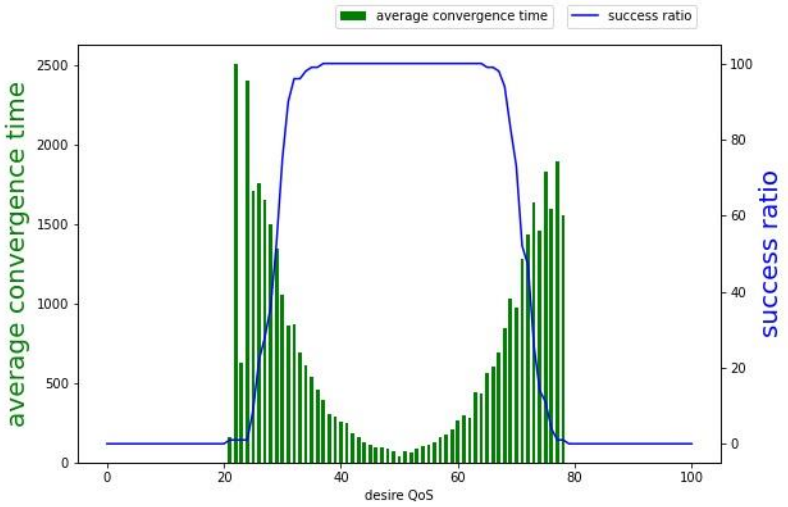
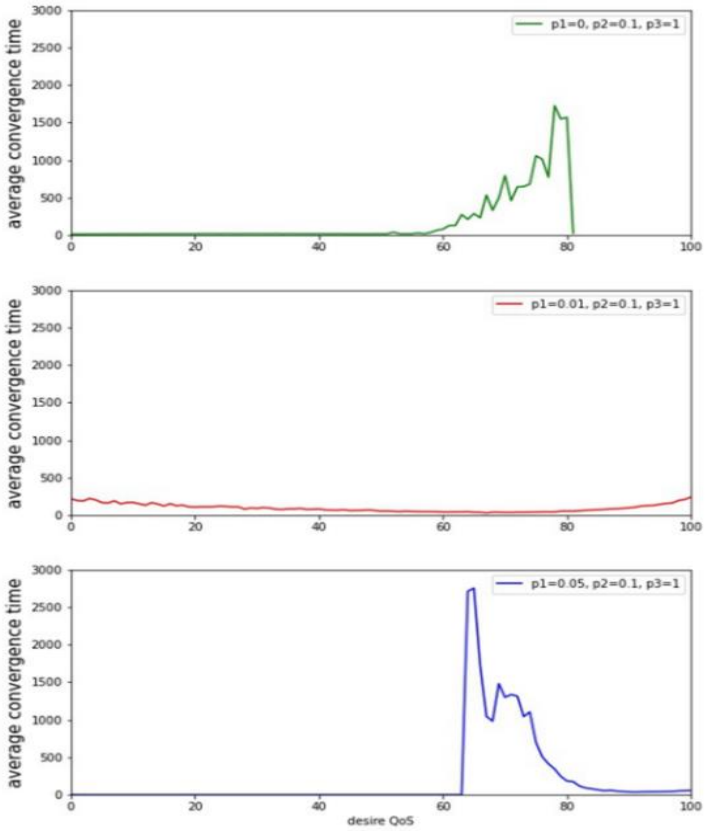


Fig. 8 Average convergence time and success rate in the Gur game approach.

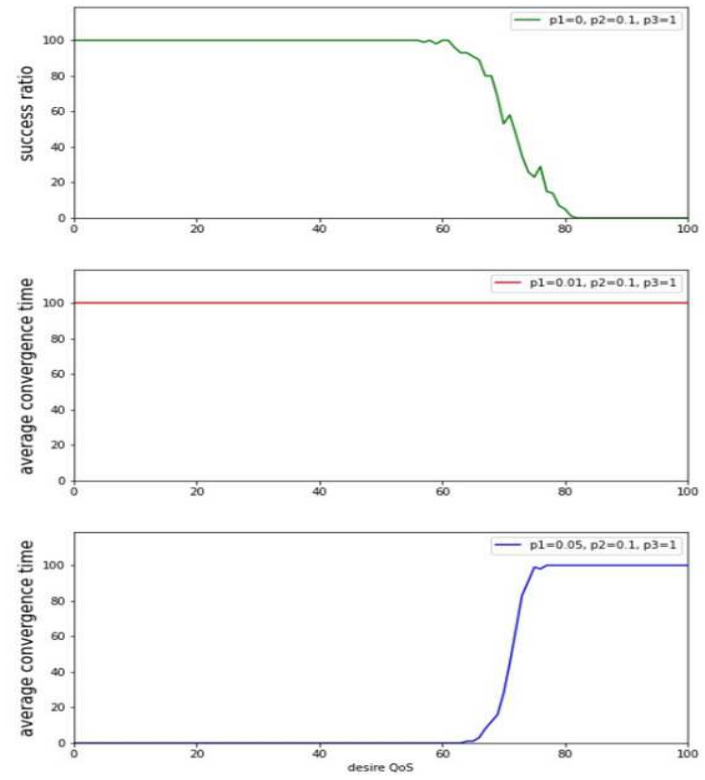
Figs. 9(a) and 9(b) show the execution of the ACK algorithm. When P_1 was set to 0.01, the success rate was 100%, and the convergence time was not more than 250 rounds. The success rate cannot reach 100 % %when P_1 is set to 0 or 0.05. Therefore, in subsequent experiments, P_1, P_2, and P_3 were set to 0.01, 0.1, and 1, respectively, for the ACK algorithm.

Figs. 10 (a) and (b) show the results of SACA, where the target value was increased from 0 to 100. We set the activation parameters to 0.001 and 0.0001, respectively, and compared the activation parameters to the sensor convergence speed impact. According to the experimental results, when the activation parameter was set to 0.0001, the convergence success rate of the SACA was close to 100%. The convergence time of SACA was shortened by approximately one-tenth that of Gur Game, which shows that SACA has strong self-organization and fast convergence speed ability.

A WSN typically consists of a large number of sensors. Therefore, when the number of sensors increases, the success rate and convergence time determine whether the system can effectively provide continuous monitoring of the environment. To further highlight the faster convergence speed of SACA compared to the ACK algorithm, we increased the total number of sensors from 1,000 to 10,000. The target values were set to 30% and 70% of the total number of sensors, respectively. When the number of active sensors reached ±3% of the target value, it indicated a successful convergence to the target value, as shown in Fig. 11.

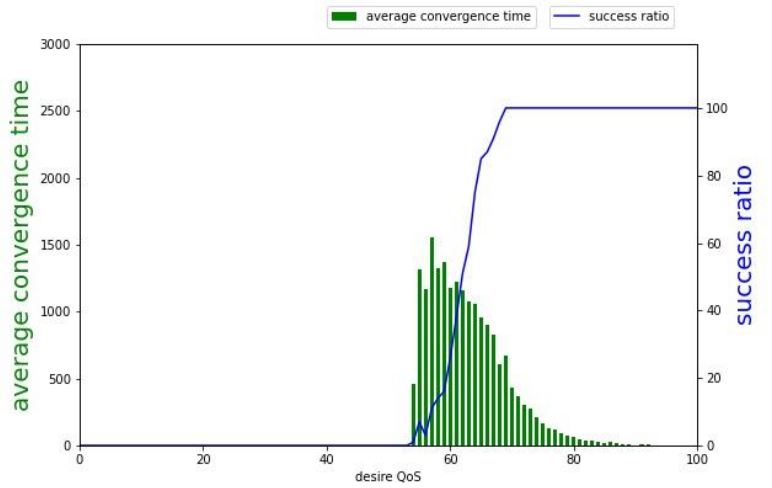


(a)

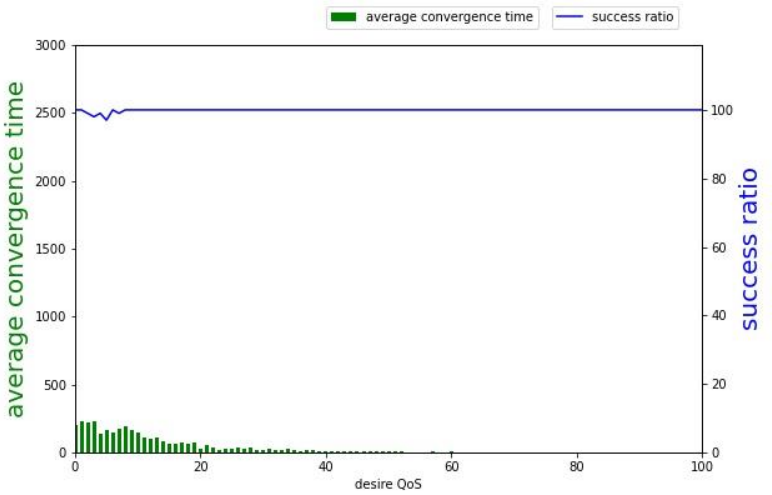


(b)

Fig. 9 (a)Average convergence time versus (b) success rate in the ACK algorithm.



(a)



(b)

Fig. 10 (a) Average convergence time and success rate with σ is 0.001 versus (b) σ is 0.0001 in the SACA.

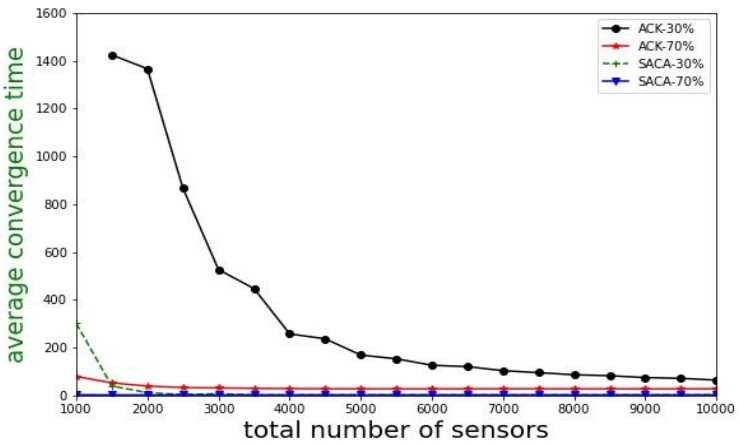


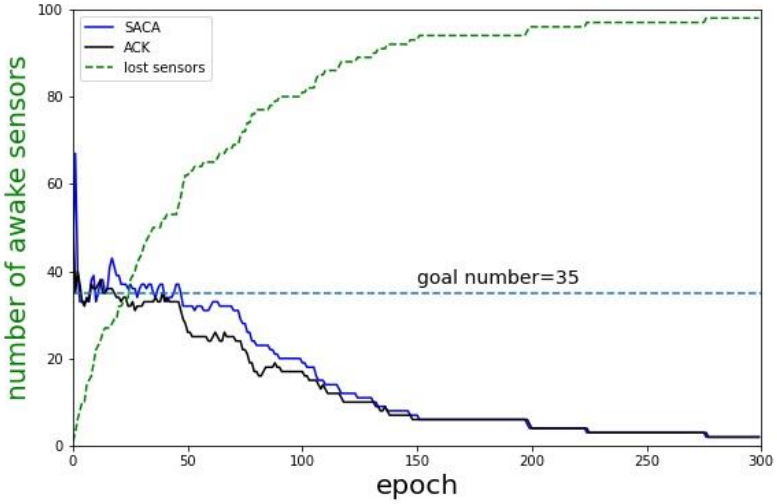
Fig. 11. Average convergence time and success rate in the Gur game approach.

5.3 Self-organization Ability Analysis with High Death Rate

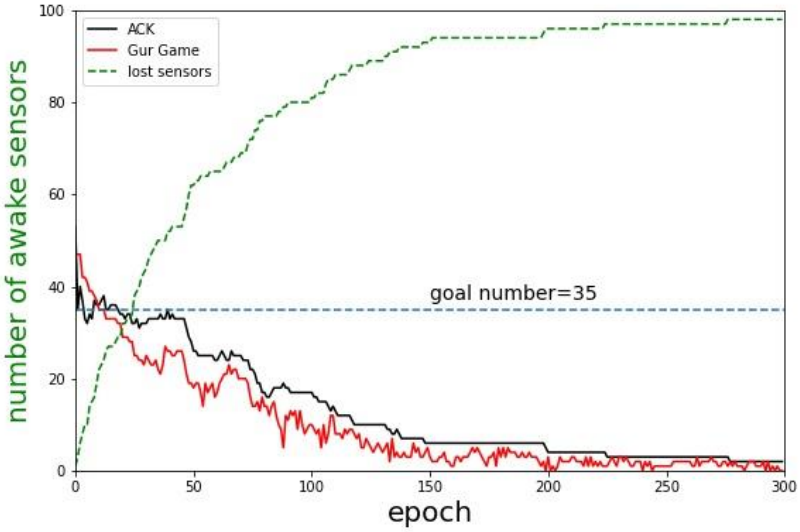
The actual network environment is often unpredictable. The sensor may be damaged or out of power, causing the total number of sensors to change. Therefore, we created a network environment with a high death rate. We verified the SACA, Gur Game, and ACK algorithms in terms of self-organization and reconstruction capabilities. As in Section 5.2, in an environment that does not consider the death or damage of the sensor, when the activation parameter σ is small, fewer sensors change from the sleep state to the active state, which makes the system more stable. However, in a network environment with a high death rate, the activation parameter σ is increased to reach the target value as soon as possible. To create a network environment with a high death rate, we generated the life span of the sensor from an exponential distribution with an average value of 70. In addition, in this experiment, the activation parameter of SACA was set to 0.01.

Fig. 12(a) shows the execution results of the SACA and ACK algorithms. The SACA algorithm took longer than the ACK algorithm to maintain the target value. Fig. 12(b) shows the execution results of ACK and Gur Game, which hardly remained at the target value. The activation parameter affects the probability of the sensor transitioning from the sleep state to the active state. In other words, if the activation parameter is higher, the sensor has a higher probability of converting its state into active state. In an environment that does not consider the death or damage of the sensor, when the activation parameter is small, fewer sensors will transition from the sleep state to the active state.

Therefore, in a network environment with small changes, we recommend setting smaller activation parameters to achieve a quick convergence to the target value. However, in a network environment with a high death rate, owing to the large changes in the network environment, a larger activation parameter should be used to increase the speed of convergence to the target value. The results show that SACA has stronger self-organization and reconstruction capabilities than Gur Game.



(a)



(b)

Fig. 12 (a) SACA and ACK versus (b) ACK and Gur game in highly death environments.

5.4 Energy-Efficiency Analysis

Numerous studies define the network life cycle as follows: the system starts execution until the first sensor death. The literature defines the network life cycle as follows: the system begins to execute until the number of active sensors cannot maintain the target value of 6. In many cases, we are concerned with the capability of the system to monitor environmental information. We adopted the latter definition in this study. The main purpose of this experiment is to illustrate the sink through individual transmission and broadcast transmission of QoS

packets, as well as to analyze these two methods in terms of the effect of power consumption on network systems.

We take the win nodes developed by the Rockwell Research Center as an example. Table 2 lists the various functions of the win nodes relative to the power consumption information. In this case, the sensor status can be divided into (a) transmission, (b) receive, and (c) idle modes. In transitions between modes, which also exhibit varying power consumption, the power consumed by the conversion between the transmission mode and the receive mode is 19.355 mW, and that consumed by the conversion between the transmission mode and the idle mode is 14.75125 mW [15]. Sensors can load battery types such as lithium ions, NiMH, NiCd, and alkaline. In this experiment, 1100 mAh NiMH batteries were used to enable the network system to provide a total power of 1131 W. The total number of sensors m was set as 100, the target value Q_0 was set to 35, and the evoke parameter σ of SACA was set to 0.001.

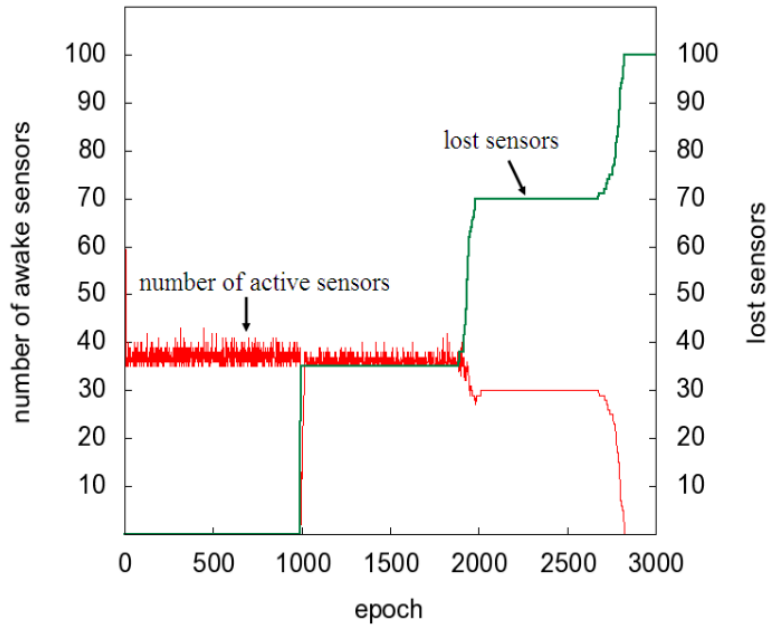
Table 2. Power consumption of Rockwell's wins nodes

State mode	MCU mode	Sensor mode	Radio mode	Power (mW)
Transmission	Active	Active	Tx, Rx	11394
Receive	Active	OFF	Rx	409
Idle	OFF	OFF	OFF	40.7

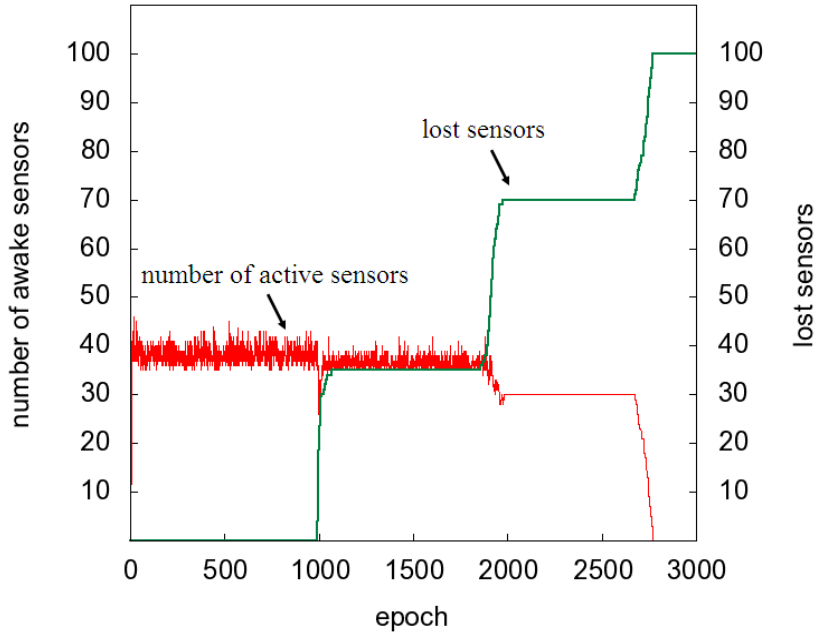
Figs. 13 (a)–(b) show the execution results of the SACA and ACK approaches. The ACK approach is unstable compared with SACA in terms of maintaining the target value, which indicates that the number of active sensors is higher.

Although the Gur game approach enables stable convergence for the target value, the number of active sensors cannot be maintained at the target value after 1,000 rounds, as shown in Fig. 14(a).

Fig. 14 (b) shows the power consumption of the three approaches. The Gur game approach is faster than the ACK and SACA approaches in terms of power consumption. The system power is exhausted by the Gur game algorithm at approximately 2,000 rounds. In the SACA and ACK approaches, the system power is exhausted at approximately 2,700 rounds.

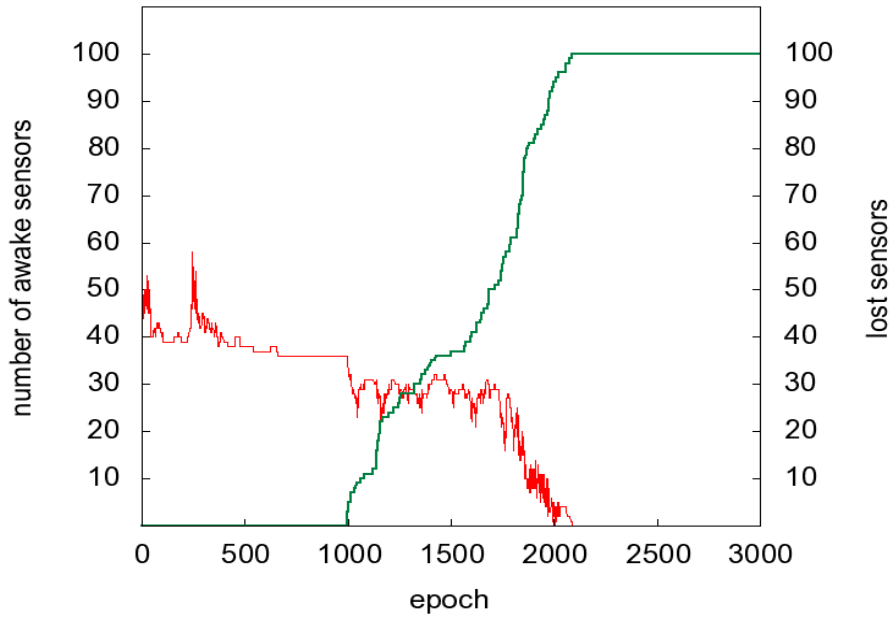


(a)

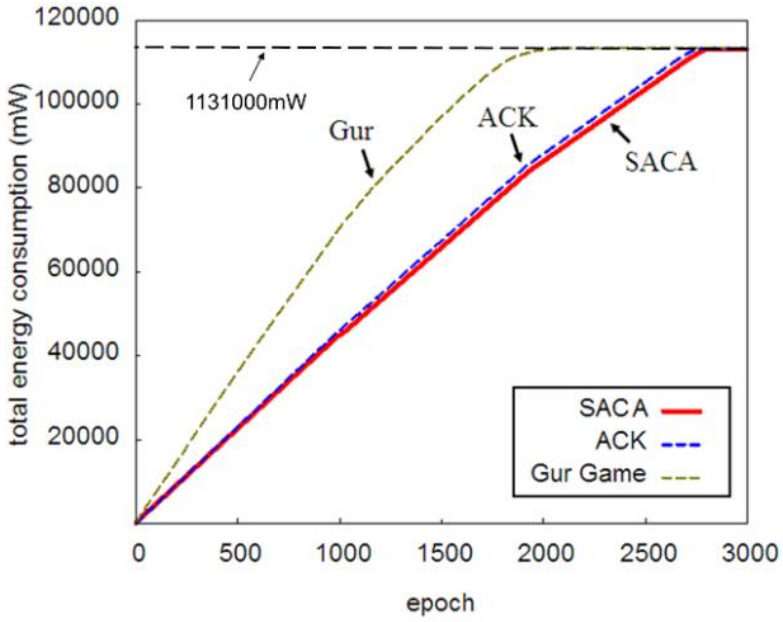


(b)

Fig. 13 Variations of the number of active sensors for the (a) SACA, (b) ACK approach.



(a)



(b)

Fig. 14 Variations of the number of active sensors for the (a) Gur game approach, (b) the total power consumption of the network system for the SACA, ACK, and Gur game.

The initial power of each sensor is E_i and the total number of sensors is m. Thus, Equation (5) can be used to calculate the total power of the entire network system as E_{max}. Equation (6) shows the ideal value E_G of power consumption for the Gur game approach, where E_T is the power consumption of the sensor in the transmission mode, E_R is the power consumption of the sensor in the receive mode, and t is the number of rounds. E_{TR} is the power consumption of the conversion of the sensors between the transmission and receive modes. Equation

(7) gives the power loss ideal value E_A of the SACA and ACK algorithms. E_{TI} refers to the power consumption of the conversion of sensors between the transmission status and idle status, where E_G and E_A are not greater than E_{max}. Equations (6) and (7) illustrate that SACA is better than the Gur game approach in terms of energy savings because sensors in sleep mode no longer receive any QoS packet from the sink. Moreover, the power consumption of the conversion of sensors between transmission status and idle status is less than that of the conversion of sensors between transmission status and receive status.

$$E_{max} = \sum_{i=1}^{m} E_i \quad (5)$$

$$E_G = [(Q_0 E_T + (m - Q_0)E_R) + E_{TR}]t, \ E_G \leq E_{max} \quad (6)$$

$$E_A = [(Q_0 E_T + (m - Q_0)E_I) + E_{TI}]t, \ E_A \leq E_{max} \quad (7)$$

6. Results and discussion

Efficient control of the number of active sensors avoids wasting power and bandwidth resources. In this study, we proposed a novel approach called sensor activity control algorithm (SACA) that applies a reward and punishment strategy to sensor nodes to establish a faster convergence time and robust self-organization. SACA not only preserves the original characteristics of the ACK algorithm, but also exhibits a faster convergence time than both the Gur game and ACK algorithms. In addition, SACA has more robust self-organization capabilities in the high death rate scenarios of sensors and is better in terms of energy efficiency compared with the Gur game and ACK algorithms.

Abbreviations

SACA: Sensor activity control algorithm; QoS: Quality of service; WSNs: Wireless sensor networks; QC²: QoS Control Scheme with Quick Convergence.

Authors' contributions

R.-G. Tsai and P.-H. Tsai conceived and designed the experiments. R.-G. Tsai performed the experiments. R.-G. Tsai and P.-H. Tsai analyzed the data and R.-G. Tsai wrote the paper.

Funding

Rong-Guei Tsai was supported by the Education Department of the Fujian Province Project, China (Grant No. JAT190580), and the Science Foundation of the Fujian Province Project, China (Grant No. 2020J01924).

Availability of data and materials

The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Ethics approval and consent to participate

Not applicable.

Competing for publication

All authors agree to submit this version and claim that no part of this manuscript has been published or submitted elsewhere.

Competing interests

The authors declare no conflict of interest.

Author details

[1] New Engineering Industry College, Putian University, Fujian 351100, China.
[2] Institute of Manufacturing Information and Systems, National Cheng Kung University, Tainan 701, Taiwan.

References

1. Akyildiz, I.F.; Su, W.; Sankarasubramaniam, Y.; Cayirci, E. Wireless Sensor Networks: a survey. Computer Networks, 2002.
2. Dubey, R.; Choubey, R.; Dubey, A.; Challenges for Quality of Service (QoS) in Wireless Sensor Networks. International Journal of Engineering Science and Technology, Vol. 2, no. 12, pp. 7395-7400, 2010.
3. Bhuyan, B.; Sarma, H.; Sarma, N.; Kar, A.; Mall, R.; Quality of Service (QoS) Provisions in Wireless Sensor Networks and Related Challenges. Wireless Sensor Network, Vol.2, no.11, pp. 861-868, 2010.
4. H.-Y. Shi,; W.-L. Wang,; N.-M. Kwok,; S.-Y. Chen,; Game Theory for Wireless Sensor Networks: A Survey. Sensors, Vol. 12, no. 7, 2012.
5. Iyer, R.; Kleinrock, L.; QoS Control for Sensor Networks. IEEE International Communication Conf. (ICC), 2003.
6. Frolik, J.; QoS Control for Random Access Wireless Sensor Networks. IEEE Wireless Communications and Networking Conf. (WCNC), 2004.
7. Kay, J.; Frolik, J.; Quality of Service Analysis and Control for Wireless Sensor Networks. IEEE International Conf., 2004.
8. Liang, B.; Frolik, J.; Sean Wang, X.; A Predictive QoS Control Strategy for Wireless Sensor Networks. IEEE International Conf. on Mobile Adhoc and Sensor Systems, 2005.
9. Kay, J. M.; Frolik, J.; An Expedient Wireless Sensor Automaton with System Scalability and Efficiency Benefits. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 2008.
10. Liang, B.; Frolik, J.; Wang, X.; Energy-efficient Dynamic Spatial Resolution Control for Wireless Sensor Clusters. International Journal of Distributed Sensor Networks, 2007.
11. Wang, H.-L.; Tsai, R.-G.; Application of Periodical Shuffle in Controlling Quality of Service in Wireless Sensor Networks. International Journal of Ad hoc, Sensor & Ubiquitous Computing (IJASUC), Vol. 4, No. 2, 2013.

12. Elshahed, E. M.; Ramadan, R. A.; Al-tabbakh, S. M.; El-zahed, H.; Modified gur game for WSNs QoS control. International Workshop on Wireless Networks and Energy Saving Techniques (WNTEST-2014), 2014.
13. Semprebom, T.; Montez, C.; de Araujo, G. M.; Portugal, P.; Skip game: an autonomic approach for QoS and energy management in IEEE 802.15.4 WSN. IEEE Symposium on Computers and Communication (ISCC), pp. 1-6, 2015.
14. Ayers, M.; Liang, Y.; Gureen Game: An Energy Efficient QoS Control Scheme for Wireless Sensor Networks. Green Computing Con-ference and Workshops, 2011.
15. Zhao, L.; Xu, C.; Xu, Y.; Li, X.; Energy-Aware QoS Control for Wireless Sensor Network. IEEE Conf. on Industrial Electronics and Applications (ICIEA), 2006.
16. Spitler, S. L.; Lee, D. C.; Integration of Explicit Effective-Bandwidth-Based QoS Routing With Best-Effort Routing. IEEE/ACM Transactions on Networking,Vol. 16, no. 4, 2008.
17. Varyani, N.; Zhang, Z.-L.; Dai, D.; QROUTE: An Efficient Quality of Service (QoS) Routing Scheme for Software-Defined Overlay Networks. IEEE Access, Vol. 8, 2020.
18. Nelakuditi, S.; Varadarajan, S.; Zhang, Z.-L.; On localized control in QoS routing. IEEE Transactions on Automatic Control, Vol. 47, no. 6, 2002.
19. Lin, S.-C.; Chen, K.-C.; Statistical QoS Control of Network Coded Multipath Routing in Large Cognitive Machine-to-Machine Networks. IEEE Internet of Things Journal, Vol. 3, no. 4, 2016.
20. Wang, H.-L.; Huang, W.-L.; QC2: A QoS Control Scheme with Quick Convergence in Wireless Sensor Networks. ISRN Sensor Networks, Vol. 2013, 2013.
21. Tsai, R.-G.; Shie, G.-R.; Tsai, P.-H.; Sensor activity control algorithm for Random Access Wireless Sensor Network Systems. International Computer Symposium (ICS), 2020.
22. Wang, H.-L.; Tsai, R.-G.; Li, L.-S.; An Enhanced Scheme in Controlling Both Coverage and Quality of Service in Wireless Sensor Networks. International Journal on Smart Sensing and Intelligent Systems, 2013.
23. Zhou, J.; Mu, C.; A Kind of Application-Specific QoS Control in Wireless Sensor Networks. IEEE International Conf. on Information Acquisition (ICIA), pp. 456-461, 2006.
24. Zhou, J.; Mu, C.; Density Dominion of QoS Control with Localized Information in Wireless Sensor Networks. International Conf. on ITS Telecommunications Proceedings, pp. 917-920, 2006.
25. Nayer, S. I.; Ali, H.; A Dynamic Energy-Aware Algorithm for Self-Optimizing Wireless Sensor Networks. In proceedings of 2008 3rd International Workshop on Self-Organizing Systems, 2008.
26. Wesam, A.; Mohammad, Q.; Orieb, A.A.; Virtual Node Schedule for Supporting QoS in Wireless Sensor Network. IEEE Jordan In-ternational Joint Conference on Electrical Engineering and Information Technology (JEEIT), 2019.

Figures

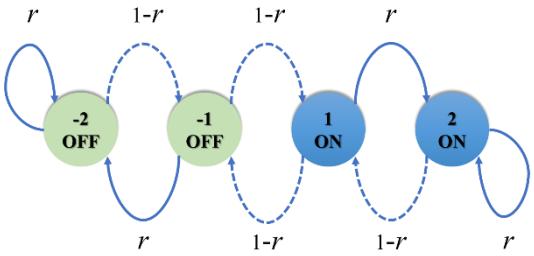


Figure 1 State machine used by the sensor in the Gur Game algorithm.

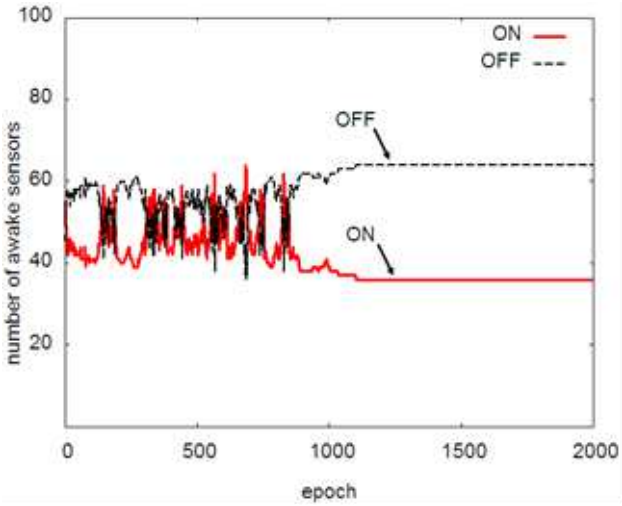


Fig. 2 Execution results of the Gur game approach when the target value Q_0 is set as 35.

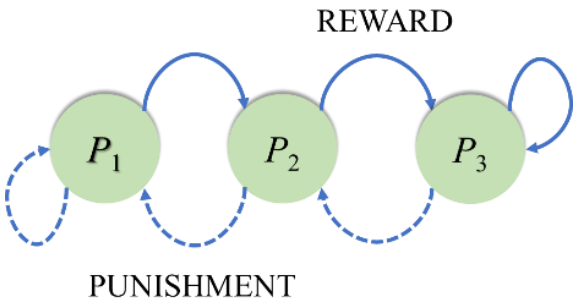


Fig. 3 Finite-state machine in the ACK algorithm

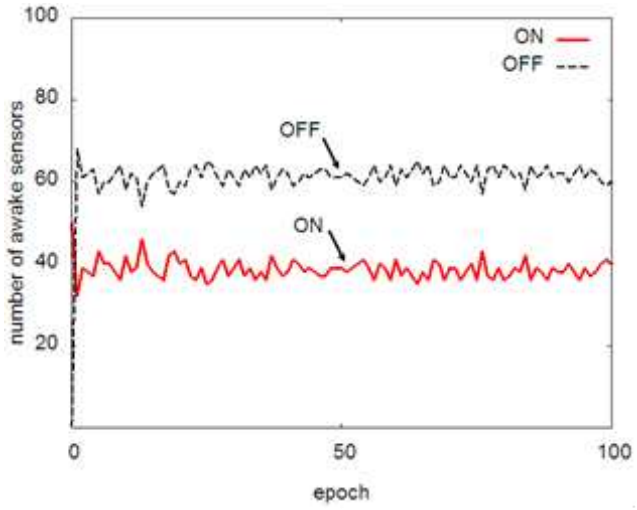


Fig. 4 Execution result of the ACK approach when the target value is set as 35.

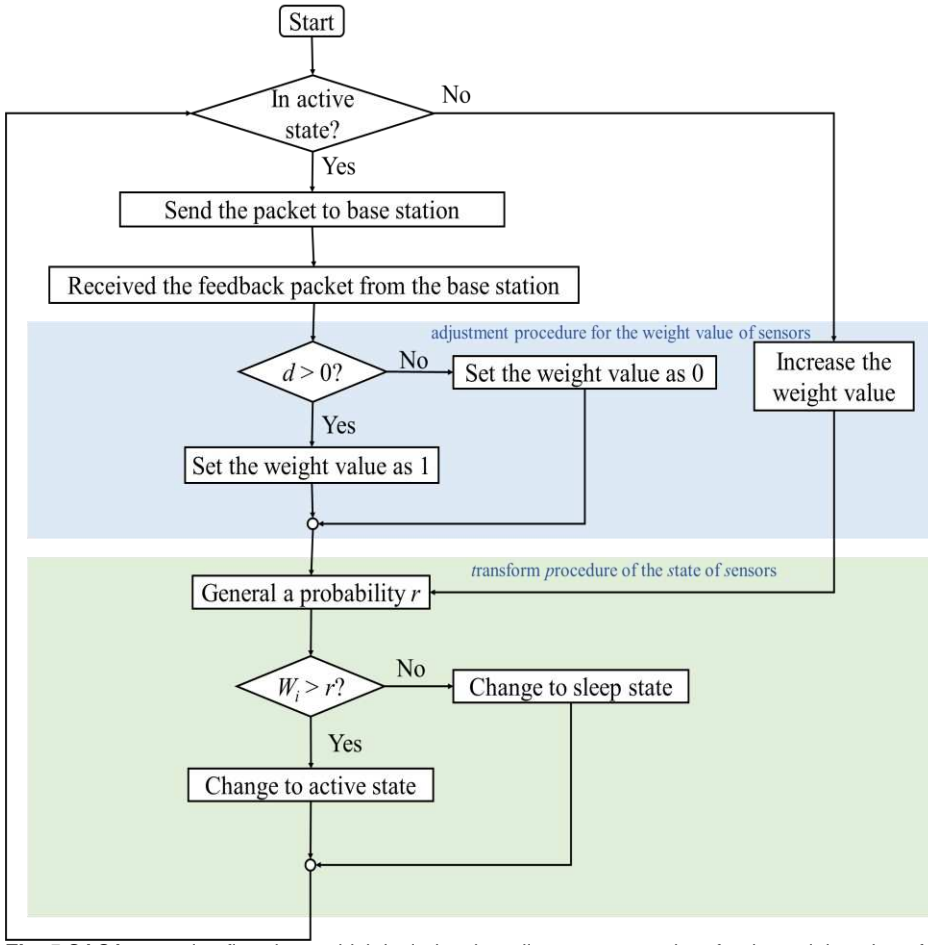


Fig. 5 SACA operation flowchart, which includes the adjustment procedure for the weight value of sensors and transform procedure of the state of sensors.

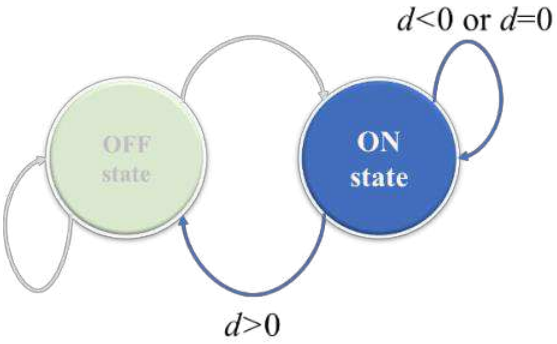


Fig. 6 The active sensor performs a state transition according to the difference value d in the feedback packet.

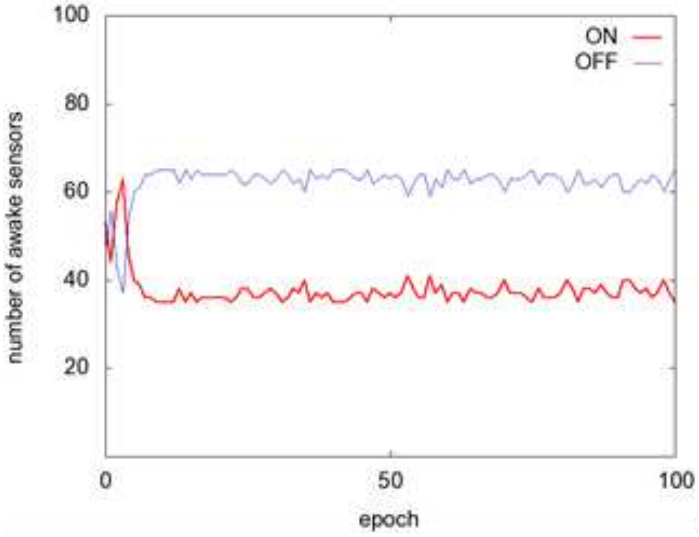


Fig. 7 Execution result of the SACA when the target value is set as 35.

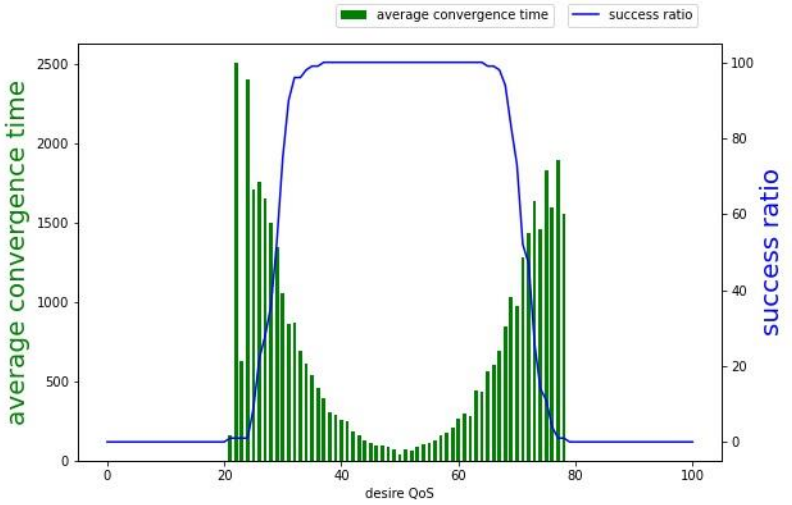
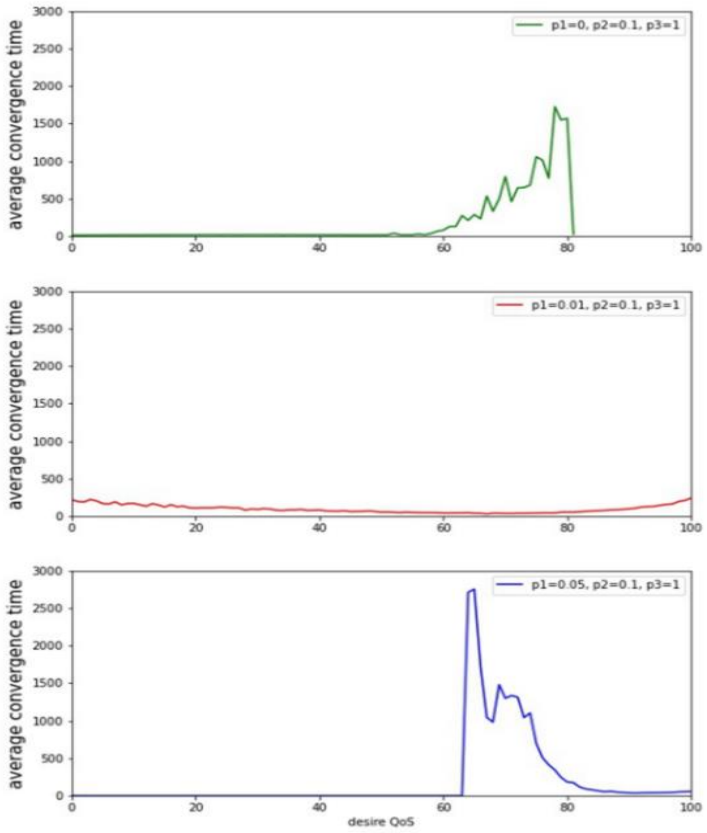
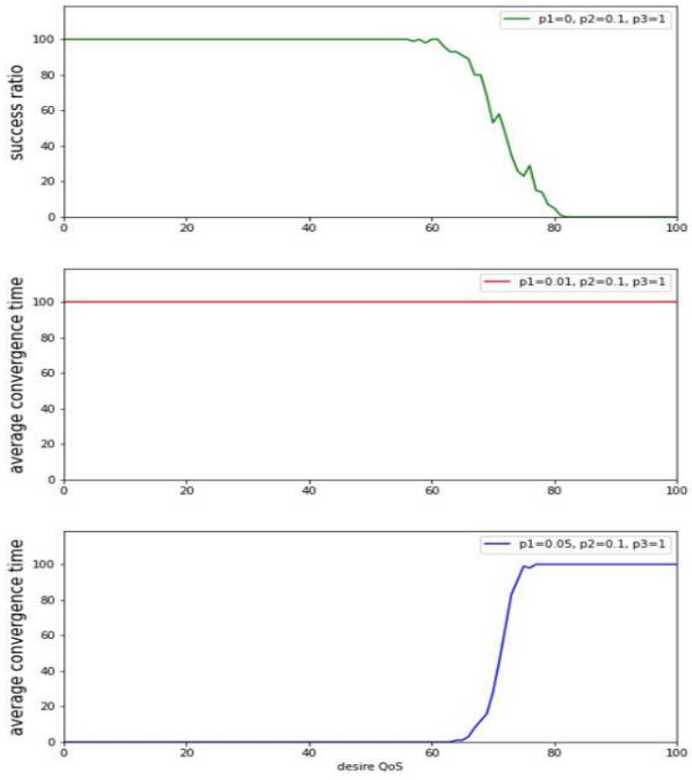


Fig. 8 Average convergence time and success rate in the Gur game approach.

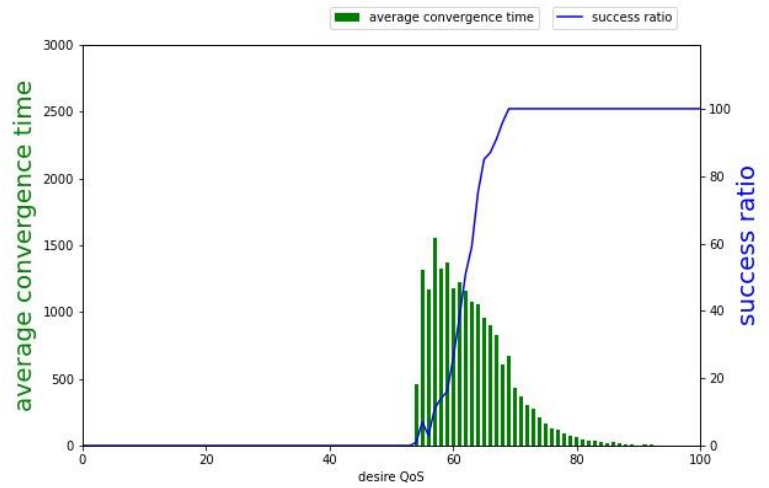


(a)

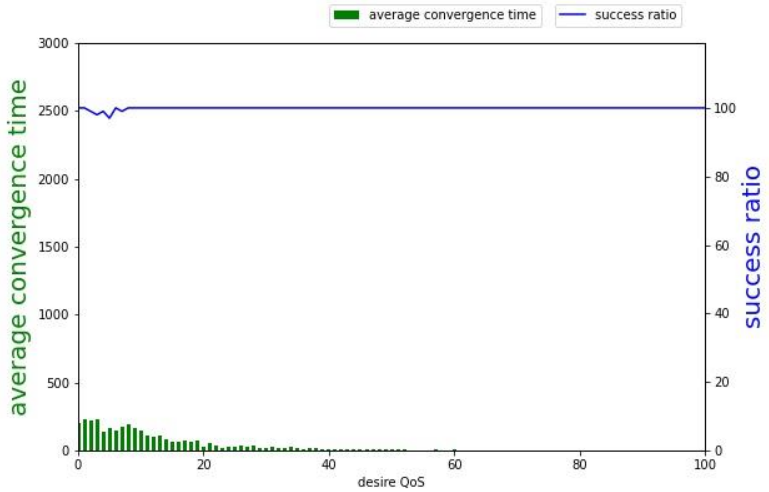


(b)

Fig. 9 (a)Average convergence time versus (b) success rate in the ACK algorithm.



(a)



(b)

Fig. 10 (a) Average convergence time and success rate with σ is 0.001 versus (b) σ is 0.0001 in the SACA.

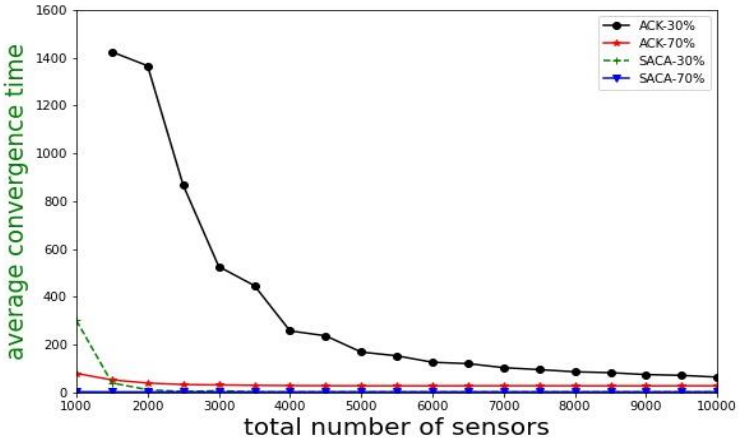
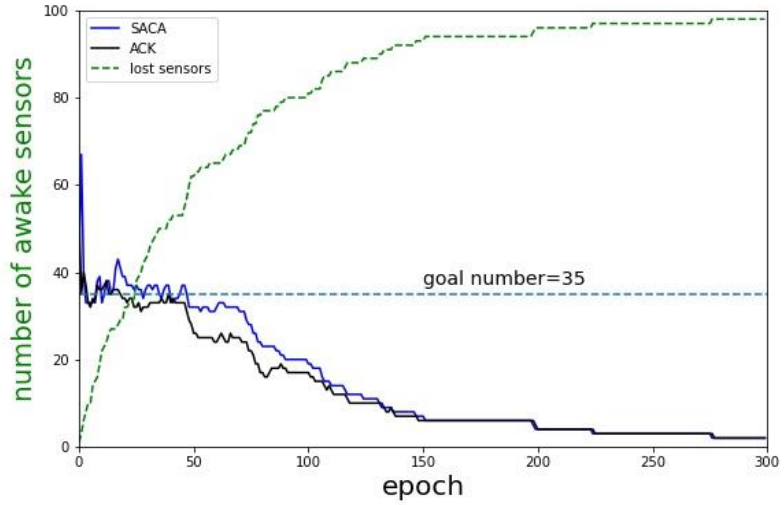
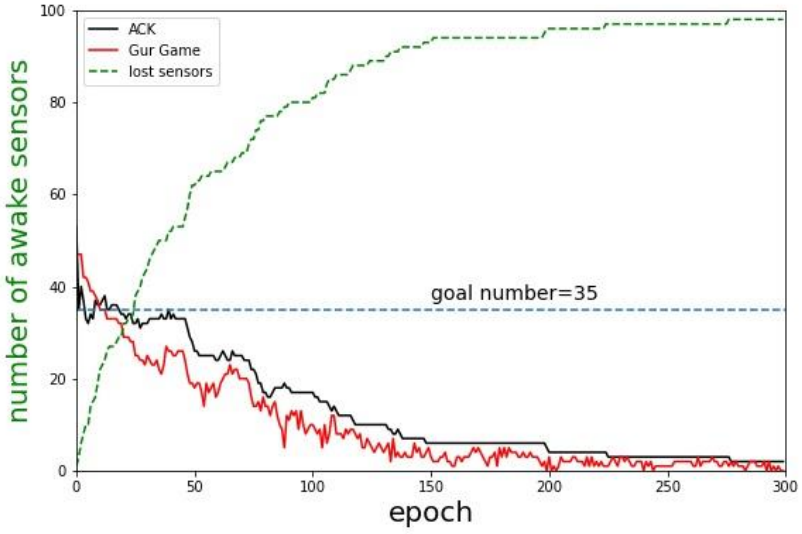


Fig. 11. Average convergence time and success rate in the Gur game approach.



(a)



(b)

Fig. 12 (a) SACA and ACK versus (b) ACK and Gur game in highly death environments.

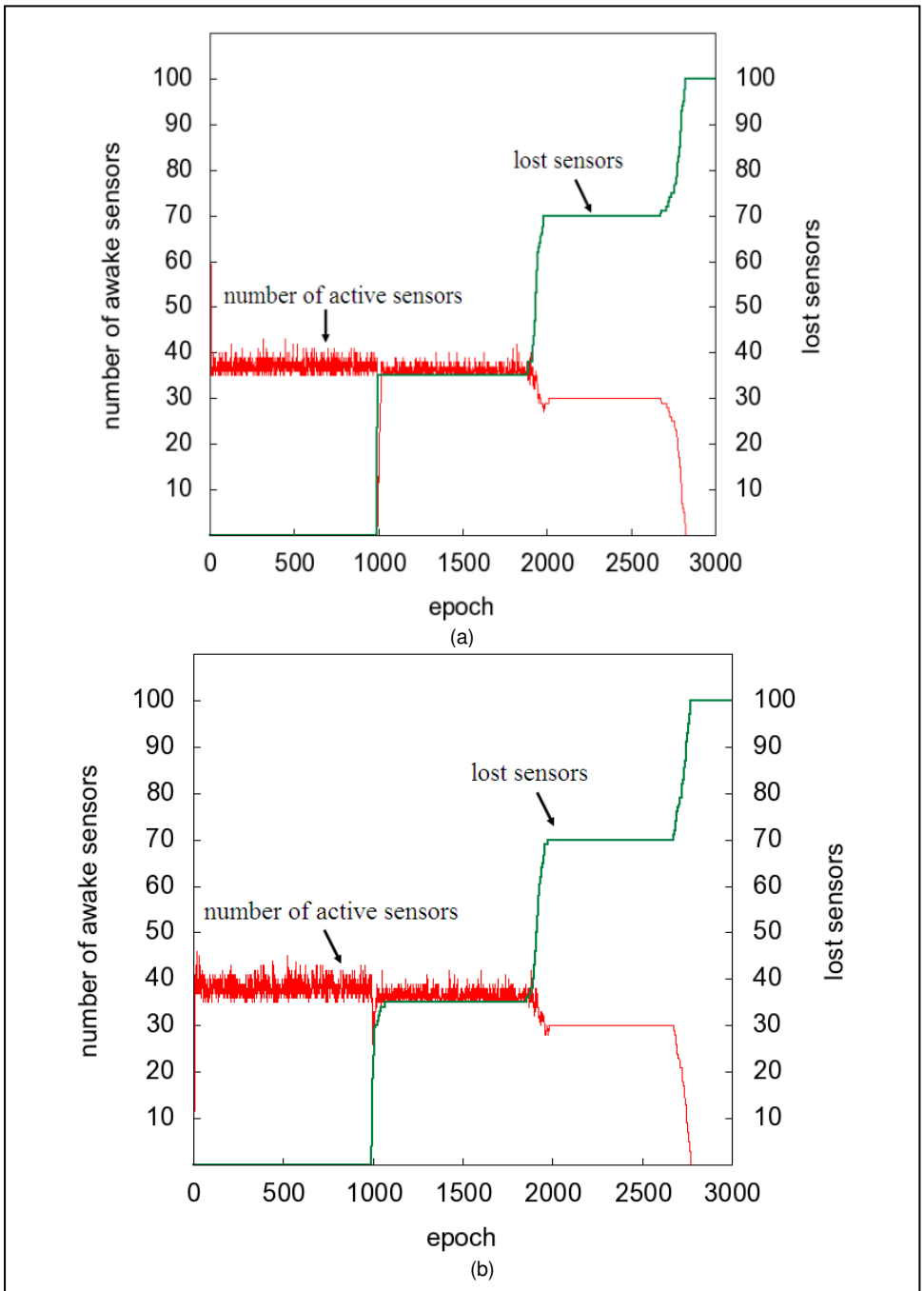
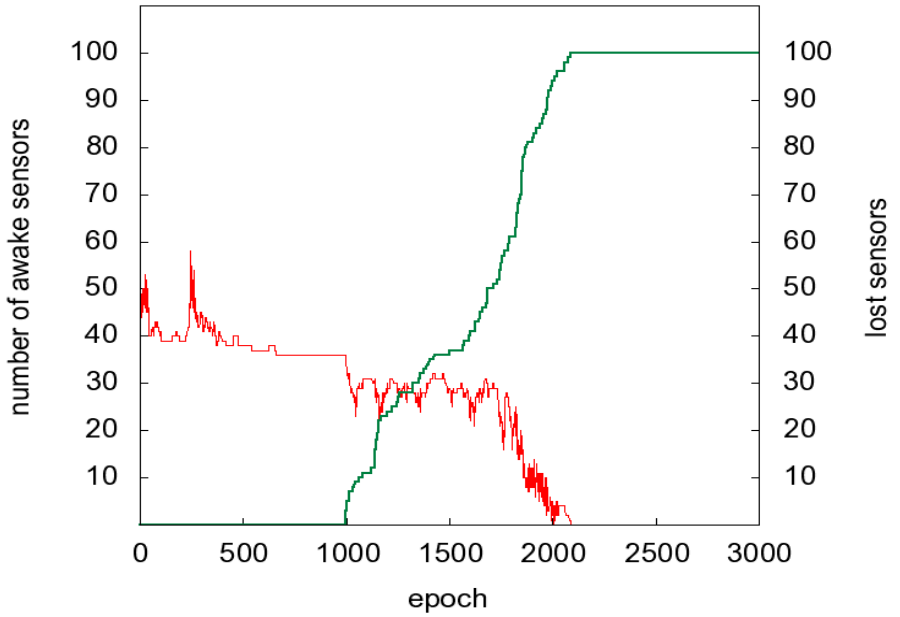
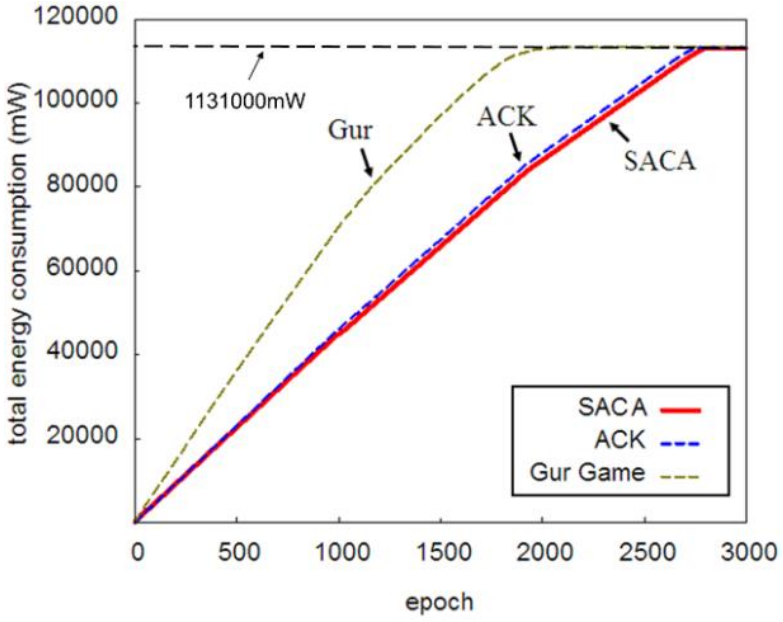


Fig. 13 Variations of the number of active sensors for the (a) SACA, (b) ACK approach.



(a)



(b)

Fig. 14 Variations of the number of active sensors for the (a) Gur game approach, (b) the total power consumption of the network system for the SACA, ACK, and Gur game.

Tables

Table 1 Experimental parameter settings

Items	**Parameter**
Total number of sensors, m	10^2, 10^3-10^4
Number of the sink	1
Target value Q_0	0-100 0.3, 0.7(Q_0/m)
Adjustment parameter, $\Delta\tau$	1
Evoke parameter, σ	0.001, 0.0001
Number of experiments	100
execute rounds for each set of experiment	3000

Table 2. Power consumption of Rockwell's wins nodes

State mode	**MCU mode**	**Sensor mode**	**Radio mode**	**Power (mW)**
Transmission	Active	Active	Tx, Rx	11394
Receive	Active	OFF	Rx	409
Idle	OFF	OFF	OFF	40.7