J Supercomput (2015) 71:1067–1094
DOI 10.1007/s11227-014-1354-z

A novel sleep scheduling scheme in green wireless sensor networks

Jing Zhang · Li Xu · Shuming Zhou · Xiucai Ye

Published online: 6 December 2014

Abstract Reduction of unnecessary energy consumption is becoming a major concern in green wireless sensor networks. Sleep scheduling is one of the efficient strategies to achieve energy saving. In this paper, we propose a novel scheme for the sleep scheduling, which is based on Decentralized Partially Observable Markov Decision Process (Dec-POMDP). A sleep scheduling algorithm with online planning (Dec-POP-SSA) with respect to Dec-POMDP is also presented. In Dec-POMDP, due to the hardness of obtaining the state spaces and the reward with mold-free environment, quasi-Monte Carlo is applied to collect state spaces such that the real-time acquisition of beliefs state is achieved, and the reward is evaluated in tracking reward and coverage connectivity intensity. Instead of producing the entire plan, Dec-POP-SSA need only find actions for the current step. We also give the theoretical analysis on the upper bound for Dec-POP-SSA. The numerical experiments show that Dec-POP-SSA may receive the highest reward.

J. Zhang · L. Xu · S. Zhou
School of Mathematics and Computer Science, Fujian Normal University, Fuzhou, China
e-mail: jing165455@126.com

S. Zhou
e-mail: zhoushuming@fjnu.edu.cn

J. Zhang
School of Information Science and Engineering, Fujian University of Technology, Fuzhou, China

L. Xu (✉)
Fujian Provincial Key Laboratory of Network Security and Cryptology, Fujian Normal University, Fuzhou, China
e-mail: xuli@fjnu.edu.cn

X. Ye
Department of Computer Science, University of Tsukuba, Tsukuba Science City, Ibaraki, Japan
e-mail: yexiucai2013@gmail.com

Keywords Wireless sensor networks · Sleep scheduling · Dec-POMDP · Quasi-Monte Carlo · Upper bound

1 Introduction

Wireless sensor networks (WSNs) have become a research hotspot recently [1]; for example, in gathering event information from animal habitats, environmental monitoring, etc. [2]. To meet the global challenge of dramatically reducing energy consumption and the exponentially growing data traffic, creating green wireless sensor networks (GWSNs) is becoming an important issue [3,4]. GWSNs normally consist of a large number of distributed sensors, each sensor is expected to work on batteries for several months to a few years without replenishing. Thus, energy efficiency becomes a critical issue in GWSNs [5–8].

In GWSNs, tracking multiple events is one of the most important researches in the last two decades [6]. Since each sensor in GWSNs has a limited range, it can detect the events, and communicate with other sensors within its communication range. To achieve the reduction of energy consumption, green networking techniques must be implemented to change the current network according to demand. Sleep scheduling is a standard approach for balancing energy consumption. By this approach, the sensors may be put into a sleep mode. It was estimated that by setting some sensors into sleep mode, more than 23 % of the total energy can be saved in a network [7]. There are two primary approaches to study the sleep scheduling problem: one approach employs duty cycle for conserving energy and extending the network lifetime, sensors stay awake only a fraction of the time, sensing and communication are done only during this awake time [5,6,8–16]. The other addresses lifetime maximization as an optimization of the energy consumption for a given network; the most commonly used optimization technique is Markov decision process (MDP) [3,17–22].

This paper attempts to examine the fundamental theory of sleeping in GWSNs for tracking, as opposed to the design of algorithm for this sleeping. An optimization problem was formulated as the form of partially observable Markov decision process (POMDP) [23], based on which, an optimal sleep scheduling algorithm with online planning (Dec-POP-SSA) is proposed to solve the sleep scheduling problem.

The main contributions of this work are listed as follows:

1. We propose a novel scheme for the sleep scheduling based on decentralized partially observable Markov decision process (Dec-POMDP). In Dec-POMDP, due to the hardness of obtaining the state spaces and the reward with mold-free environment, quasi-Monte Carlo (QMC) is applied to collect state spaces such that the real-time acquisition of beliefs state is achieved, and the reward is evaluated by tracking reward and coverage connectivity intensity.
2. A sleep scheduling algorithm with online planning (Dec-POP-SSA) and Dec-POMDP is presented. Instead of producing the entire plan, Dec-POP-SSA only finds actions for the current step. Those actions are kept in the belief pools. Cluster head maintains a sub-belief pool, and every sensor in the same cluster collaborates to reduce network traffic through selective communication.

The remainder of this paper is organized as follows: Sect. 2 introduces the related works. Section 3 presents the problem statement of sleep scheduling. Section 4 is devoted to the Dec-POP-SSA algorithm. The theoretical and numerical analyses are shown in Sects. 5 and 6, respectively. Finally, Sect. 7 concludes this paper.

2 Related works

There are two primary approaches to explore the sleep scheduling problem in earlier literature: The first one is to design a sleep scheduling method with the duty cycle scheme. The second one is to utilize the technique of optimization. For the first one, Zhao et al. [9] present the virtual backbone scheduling (VBS) for WSNs. To prolong the network lifetime, it forms multiple, overlapped backbones to work alternatively. Nan et al. [10] present a coverage-guaranteed sleep/wake scheduling scheme (CDSWS). Liu et al. [11] propose a randomized scheduling scheme with guaranteed connectivity (RSGC) for WSNs, which takes into consideration the maintenance of both coverage ratio and connectivity for each group. Wang et al. [12] give a structure named two-hops-cluster scheme, which also considers the maintenance of covered and connected in the network. Ding et al. [13] establish an adaptive partitioning scheme called connectivity-based partition approach (CPA), which divides the network into several groups, while only one node in each group will be selected to be active to form a backbone network. CPA ensures the effective connectivity of the network. Yu et al. [14] propose a distributed nucleus algorithm (DNA) for domatic partition, which considers the coverage guaranteed. Kim et al. [6] address the directional cover and transmission problem of organizing the directional sensors into a group of non-disjoint subsets to extend the network lifetime. Jabbar et al. [5] propose one cluster-designing algorithm for lifetime improvement of wireless sensor networks. Censor-Hillel et al. [15] give a scheme based on network connectivity. Zhang and Li [16] focus on employing different scheduling strategies on border nodes and internal nodes, which also guarantee the coverage. Furthermore, the duty cycle scheme can use the powerful tools, such as queueing theory, game theory and the parallelization methodologies [24–33].

It is easy to see that the duty cycle approach aims to ensure coverage and connectivity. In every time slot, some sensors need to be woken up, and each sensor is awake for a predetermined time slot. Furthermore, it does not use information about the location of the events that happened in the network. Compared with the duty cycle approach, the second approach addresses sleep scheduling as technique of POMDP, which depends on the location information of the events. These approaches will result in a tradeoff curve between tracking reward and energy consumption, which can save more energy than that by the duty cycle approaches.

Fuemmeler and Veeravalli [17] identify two suboptimal sleeping policies, the first cost reduction (FCR) solution and the Q-cost-based MDP (QMDP) solution, which will be applied in optimizing the tradeoff between energy cost and tracking error without assuming the use of waking up radios. They assume that the sleeping sensors can not be woken up externally, then, internal timers need to be set when the sensors can wake up. Therefore, the control actions correspond to the sleep durations of awake sensors. Atia et al. [20,21] study a single object tracking problem where the sensors

Table 1 Sleep scheduling algorithms in WSNs

References	Goal	Primary approach	Solution
Liu et al. [11]	Coverage guaranteed	Duty cycled	–
Ding et al. [13]	Coverage guaranteed	Duty cycled	–
Nan et al. [10]	Coverage guaranteed	Duty cycled	–
Wang et al. [12]	Coverage guaranteed	Duty cycled	–
Zhao et al. [9]	Coverage guaranteed	Duty cycled	CDP
Jabbar et al. [5]	Coverage guaranteed	Duty cycled	cover tree
Yu et al. [14]	Coverage guaranteed	Duty cycled	DP
Kim et al. [6]	Coverage guaranteed	–	Cluster
Censor-Hillel et al. [15]	Coverage guaranteed	Duty cycled	CDP
Zhang et al. [16]	Coverage guaranteed	Duty cycled	–
Fuemmeler and Veeravalli [17]	Object tracking	POMDP	Monte Carlo simulation
Fuemmeler and Veeravalli [18]	Object tracking	POMDP	Monte Carlo simulation
Fuemmeler et al. [19]	Object tracking	POMDP	Monte Carlo simulation
Atia et al. [20]	Object tracking	POMDP	–
Jiang et al. [21]	Object tracking	POMDP	–
Mihaylov et al. [22]	Object tracking	POMDP	Reinforcement learning
Dec-POP-SSA	object tracking	Dec-POMDP	Quasi-Monte Carlo simulation

can be turned on or off at consecutive time slots to conserve energy based on the framework of a POMDP. Mihaylov et al. [22] present a decentralized reinforcement learning (RL) approach for self-organizing wake-up scheduling in wireless sensor networks. Fuemmeler et al. [19] establish the approach which builds on the policies designed in [17]; they use Monte Carlo simulation and learning algorithms to compute those parameters, for the special case of a discrete state space with continuous Gaussian observations. They also prove that the POMDP approach can result in better asymptotic behavior as the size of the network becomes large. Fuemmeler and Veeravalli extend the results to multi-object tracking in [18].

A bright full view of the improvement is shown in Table 1, it is clear that the coverage can be guaranteed by the duty cycle approaches. However, sleep scheduling scheme through technique of POMDP is more effective for event tracking.

3 Problem formulation

3.1 Background of Dec-POMDP

Markov decision process (MDP) provides a unified model representation for planning under uncertainty environment [23]. The key decision factor in MDP is the state of the system, which contains all the information the agent decision needs for the current strategy. That is to say, according to the current state, agent can make optimal decisions, without having to consider other information. This is the Markov property of MDP. In

the sleep schedule problems, the sensor does not have direct access to the system status information; the local feedback information of state comes from local information, which is called observation. Partially observable Markov decision process (POMDP) extends the MDP model to handle state's uncertainty by incorporating observations and a probabilistic model of their occurrence, which is more suitable for sleep schedule problems in GWSNs. The probabilistic model of the occurrence is denoted as belief state.

Different formal models for POMDP policy have been proposed over the last decade. The main bottleneck is the famous curse of dimension. Sleep schedule problems in GWSNs also have such problems. First, in addition to the sensors' states, the observed sequence of steps with the decision always shows exponential growth. Second, each individual sensor may have different partial information about other sensors and the state of the whole network. Since only by this way, it can coordinate with each other between multi-nodes, then cooperate to complete common sleep schedule tasks. However, it is sometimes difficult to evaluate the state of the sensor quantitatively, especially under uncertainty environment. Dec-POMDP is a mathematical framework, which is useful for modeling resource control problems where the decision making is decentralized [34]. Dec-POMDP offers a general approach to deal with resource management in a multi-node scenario with decentralized control, whose goal is to solve the problem of common multi-node teamwork.

In cooperative decision-making settings such as sleep schedule problems, the sensors just have to wake up with a plan that maximizes the expected reward of the network. The planning in Dec-POMDP takes place either offline or online. In offline planning, sleep scheduling plans are distributed to each sensor and executed by each sensor based on its information, and the whole plan needs to be generated at once. Hence the planning phase can be centralized as long as the execution is decentralized. However, online planning algorithms often interleave planning with execution, the current step is only needed for finding sensors' states. Sensors can share information with each other by communication.

Since a very limited time is needed to consider distributed coordination among the sensors in online planning, thus the online planning algorithms usually combine with some offline algorithms. In the offline programming, the form of policy is often represented as one full tree, the target of each iteration step in the sleep scheduling is to choose optimal subtree for each branch. While in the online planning, historical information of the sensors' policy is expressed as series form, the goal of such planning is to select an optimal operation for each of the historical information. In this paper, the Dec-POP-SSA algorithm aims to solve the coordination between sensors and the efficient utilization of limited resources by online planning. The key of the algorithm is to let each sensor calculate the joint policy independently, and to improve decision making reward maximum.

3.2 Formal model of Dec-POMDP

Some notations that will be used throughout this paper are given first, the important symbols with their definitions are collected in Table 2.

Table 2 Notation

Symbols	Definitions
n	The number of the sensor in the network
I	A finite set of agents indexed $1, \ldots, n$
$S : s_i$	A finite set of system states
$A_i : a_i$	A finite set of actions available to sensor i
Ω_i	A finite set of observations available to sensor i
P	A state transition probability table
$O : o_i$	A table of observation probabilities
R	A reward function
$\overrightarrow{a}$	A joint action
$\overrightarrow{o}$	A joint observation
b^0	The initial belief state distribution
u_l^k	The sleep time supplied to sensor l at time k
μ_k	The policy at time k
r_l^k	The sleep timer of sensor l at time k
L_k	The events occurs at time k
λ_k	The central unit at time k
c	Unit energy cost
Cov	Coverage intensity
Con	Connectivity intensity
γ_1, γ_2	The discount factors
h_i^t	A set of histories for node i
$p_{(x)}^t(b)$	A set of joint belief states of cluster (x) at time t
$\theta_{(i)}$	The history index for node i
$b_{(x)}^t$	The joint belief induced for the joint history
$\lvert C_{(x)} \rvert$	The number of nodes in the cluster (x)
Z_j	The 'bits' of n in base q
ε_i, δ	Threshold for Dce-POP-SSA
$\mathcal{K}$	The minimum number of sampling
$\mathcal{T}$	The number of the policy steps
$\mathbb{1}$	The indicator function

Definition 1 (*Dec-POMDP*) [23] A decentralized partially observable Markov decision process is a tuple

$$\langle I, S, \{A_i\}, \{\Omega_i\}, P, O, R, b^0 \rangle,$$

where, $\overrightarrow{A} = \times_{i \in I} A_i$ is the set of joint actions, $a_i \in A_i$, and $\overrightarrow{a} = \langle a_1, \ldots, a_n \rangle$ denotes a joint action. $\overrightarrow{\Omega} = \times_{i \in I} \Omega_i$ is the set of joint observations, $O_i \in \Omega_i$, and $\overrightarrow{o} = \langle o_1, \ldots, o_n \rangle$ denotes a joint observation. $P(s'|s, \overrightarrow{a})$ denotes the probability that taking joint action $\overrightarrow{a}$ in state s results in a transition to state s'. P describes the

stochastic influence of actions on the environment. $O(s'|s, \overrightarrow{a})$ denotes the probability of observing joint observation $\overrightarrow{o}$ after taking joint action $\overrightarrow{a}$ and reaching state s'. O explains how the sensors perceive the state of the environment.

For sleep scheduling problem, each sensor can be in one or two states: awake or asleep. One sensor in the awake state consumes more energy than that in sleep state. However, event sensing can be performed only when the sensors are in the awake state. An awake sensor only detects whether one or more events are within its range, and can not detect the exact number of events present or which events are present. It is presumed that during the following actions are taken, sensors wake up and remain the awake state for one time unit.

1. The sensor sends a binary observation to the central unit that indicates whether one or more events are within its range;
2. The sensor receives a new sleep time from the central controller.

Since it is impractical for the wake-up signals, a timer is initialized at the sensor that is decremented by one time unit each time step, the sensor wakes up until the timer expires, this is the only mechanism for waking up one sensor. A state summarizes all the relevant features of the dynamical system and satisfies the Markov property. That is, the probability of the next state depends only on the current state and the joint action, not on the previous states and joint actions:

$$P(s^{t+1}|s^0, \overrightarrow{a}^0, \ldots, s^{t-1}, \overrightarrow{a}^{t-1}, s^t, \overrightarrow{a}^t) = P(s^{t+1}|s^t, \overrightarrow{a}^t). \tag{1}$$

Let u_l^k denote the sleep time input supplied to sensor l at time k, which can be rewritten as

$$u_l^k = \mu_k(p_l^k, r_l^k). \tag{2}$$

The value of the sleep timer of sensor l at time k is denoted as r_l^k, the evolution of the residual sleep time can be described as

$$r_l^{k+1} = (r_l^k - 1)\mathbb{1}_{\{r_l^k > 0\}} + u_l^k \mathbb{1}_{\{r_l^k = 0\}}. \tag{3}$$

There are n sensors in the network, for each time slot k and each sensor $l \in \{1, \ldots, n\}$. The Eq. (3) is composed of two parts. The first one implies that if the sensor is currently asleep, which is denoted as $\mathbb{1}_{\{r_l^k > 0\}}$, then the sleep timer is decremented by 1. The second one implies that if the sensor is currently awake, which is denoted as $\mathbb{1}_{\{r_l^k = 0\}}$, then the sleep timer is reset to the current sleep time input for that sensor.

Denote L_k the events occur at the range of some sensors at time slot k. Unfortunately, L_k is recognized only when the events locations are being tracked precisely. Thus, there is a dynamical system with partially observed state information. If the observation available to the central unit at time k is denoted by

$$\lambda_k = \begin{bmatrix} p_1^k, r_1^k \\ \ldots \\ p_n^k, r_n^k \end{bmatrix}, \tag{4}$$

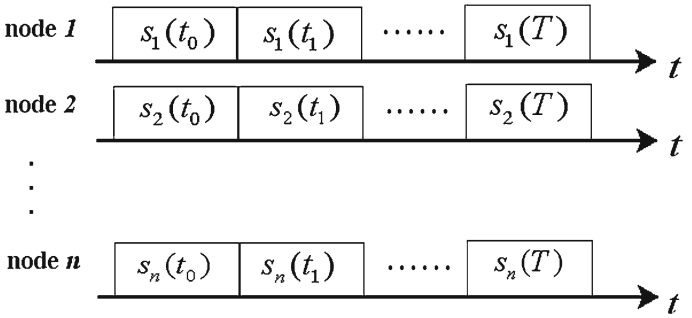

Fig. 1 The status of every nodes in the network

where p_l^k is a vector of observations with

$$p_l^k = p_l(b^k = b). \tag{5}$$

To provide a method for centralized control, the presence of an extra is assumed as a central controller. The central controller keeps track of the state of the network and assigns sleep times to sensors that are awake. After initial distribution of sleep time for every sensor, the centralized control wants to maintain the belief pool. Each sensor computing policy is distributed. The state of sensor information is referred to as sensors status history, it is the sequence S that includes state behavior λ_k, reward R, observation O, feedback response and so on. In detail, $S = \{s_i | i = 1, 2 \ldots, n\}$, where $s_i = \{(\lambda, R, O, \ldots)_1, \ldots (\lambda, R, O, \ldots)_t\}$. Within a certain time slot, in the whole network, the possibility of environmental states are

$$S_i(t) = s_1(t)s_2(t) \cdots s_n(t), s_j(t) \in \{0, 1\},$$

where $i = 1, 2, \ldots, M$; $j = 1, 2, \ldots, n$, $M = 2^n$. All possible sensor status is $S(t) = [S_1(t), \ldots, S_M(t)]$, which is shown in Fig. 1.

Theorem 1 *There are n sensors in the network, $\lambda = [\lambda_1, \ldots, \lambda_{\mathcal{T}}]$ is the Sufficient Statistic of the states $S_i(t)$, where λ_t is defined as formula (4).*

Proof At time t, the conditional probability of the state $S_i(t)$ at time $t - 1$ is

$$\begin{aligned} Pr[S(t) = S_i(t)|\lambda] &= Pr[s_1(t) = j_1, \ldots, s_n(t) = j_n | s_1(t-1) \\ &= \lambda_1, \ldots, s_n(t-1) = \lambda_n], i \in [1, M]. \end{aligned} \tag{6}$$

Since the nodes are independent to each other, we have

$$\begin{aligned} &Pr[S(t) = S_i(t)|\lambda] \\ &= Pr[s_1(t) = j_1 | s_2(t) = j_2, \ldots, s_n(t) = j_n, s_1(t-1) = \lambda_1, \ldots, s_n(t-1) = \lambda_n] \\ &\times Pr[s_2(t) = j_2, \ldots, s_n(t) = j_n | s_1(t-1) = \lambda_1, \ldots, s_n(t-1) = \lambda_n] \end{aligned}$$

$$= \prod_{k=1}^{n} Pr[s_k(t) = j_k | s_k(t-1) = \lambda_k]. \tag{7}$$

Thus, the state $S_i(t)$ can be obtained by λ, that is to say, $\lambda = [\lambda_1, \ldots, \lambda_{\mathcal{T}}]$ is the Sufficient Statistic of the states $S_i(t)$. □

Under initialization of the belief stat b^0, the evolution of $p^k = [p_1^k, \ldots, p_n^k]$ is difficult to be expressed mathematically, but it is a standard nonlinear filtering operation. The quasi-Monte Carlo sampling will be used to obtain p^k, which will be described in Sect. 4 in detail.

3.3 Reward of sleep schedule problem

The reward present in the sleep scheduling problem will be identified in this section, which is decomposed into three components: the first reward is the profit for the coverage and connectivity, the second is the tracking reward, and the third is an energy cost of $c > 0$ for each sensor that is awake. This model is used for simplicity, although it would perhaps take different cost at each sensor, the energy cost can be written as $\sum_{l=1}^{n} c\mathbb{1}_{\{r_l^k=0\}}$.

Definition 2 (*coverage intensity*) Suppose that there are n sensors $\{v_1, v_2, \ldots, v_n\}$ and m events $\{e_1, e_2, \ldots, e_m\}$ in the network. The network coverage intensity Cov is the ratio of the time when an event happens in the field of the sensor network is covered by at least one active sensor node to the total time.

If these k events are covered by the active sensor v_i, which is defined as $cv_i = k$, for any active sensors $v_i \in V$, the coverage intensity can be calculated as $\mathrm{Cov}(v_i) = \frac{cv_i}{m}$ if the area is covered by multiple active sensors $v_i, i = 1, 2, \ldots, n$. Then the coverage intensity can be calculated as:

$$\mathrm{Cov} = E\left(\mathrm{Cov}\left(\bigcup_{v_i \in \mathbb{1}_{\{r_k=0\}}} v_i\right)\right) = E\left(\sum_{v_i \in \mathbb{1}_{\{r_k=0\}}} \frac{cv_i}{m}\right). \tag{8}$$

Definition 3 (*Connectivity intensity*) Let A and B be two different clusters. For any sensor u in A and any sensor v in B, clusters A and B are called neighboring clusters if $d(u, v) \leq \sqrt{3}|\mathrm{radius}|$. The connection value between u and v is referred to be 1, say $\mathrm{Con}_{uv} = 1$ if both of u and v are the active sensors and at least one active sensor between them; otherwise, $\mathrm{Con}_{uv} = 0$. Thereby, the connectivity intensity can be calculated as:

$$\mathrm{Con} = \sum_{u_i \in \mathbb{1}_{\{r_k=0\}}} \sum_{v_j \in \mathbb{1}_{\{r_k=0\}}} \frac{2\mathrm{Con}_{u_i v_j}}{|C_{(A)}| + |C_{(B)}|}, \tag{9}$$

where $i \in \{1, \ldots, |C_{(A)}|\}$, $j \in \{1, \ldots, |C_{(B)}|\}$.

The total reward for time slot k can be expressed as

$$R\left(p^k, r_k\right) = \sum p^k \Bigg(\sum_{l=1}^{n} \mathbb{1}_{\{r_l^k=0\}} \gamma_1 (\text{Cov} + \text{Con}) + \gamma_2 \Bigg(\mathbb{1}_{\{r_k=0\}} e_k - \mathbb{1}_{\{r_k>0\}} o_k - \sum_{l=1}^{n} c \mathbb{1}_{\{r_l^k=0\}} \Bigg) \Bigg). \tag{10}$$

Tracking Reward is defined as: $TR = \mathbb{1}_{\{r_k=0\}} e_k - \mathbb{1}_{\{r_k>0\}} o_k$, which can be further decomposed into two components. The first component is tracking right when the sensors are woken up and observe some events (e_k) that occur. The second component is the observation error that occurs when there is a failure to observe particular events (o_k), both of γ_1 and γ_2 are the discount factors.

The infinite horizon reward for the system is given by

$$R\left(p^0, r_0, \mu_0, \mu_1, \ldots, \mu_T\right) = E\left[\sum_{k=1}^{I} R\left(p^k, r_k\right)\right], \tag{11}$$

The goal is to compute the solution to

$$R^*\left(p^0, r_0\right) = \max_{\mu_0, \mu_1, \ldots, \mu_T} R\left(p^0, r_0, \mu_0, \mu_1, \ldots, \mu_{\mathcal{T}}\right). \tag{12}$$

4 Suboptimal solutions

However, since an optimal solution could not be found for the sleep scheduling problem, this paper turns attention to find suboptimal solutions to the sleep scheduling problem.

Each sensor selects actions individually by Dec-POMDP policy, but it is the resulting joint action that produces the outcome. Coordination is thus a key aspect in such systems. The goal of coordination is to make sure that the individual decisions of the sensors results in optimal decisions for the cluster as a whole. The joint policy is calculated on real-time online. As long as all the sensors in the same cluster maintain the constant sub-belief pool, the outcome plans they compute will be the same. This guarantees that the strategies of the sleep scheduling are coordinated.

Definition 4 (*Belief pool*) [23] A belief pool at time slot t is defined by a tuple $< \{h_{(x)}^t | i \in I\}, p_{(x)}^t >$, where h_i^t is a set of histories for sensor i and p^t is a set of joint belief states.

The whole belief pool is partitioned by clusters to store information, each sub-belief pool stores the states of sensors, which are in the same cluster.

This section introduces a new algorithm, Dec-POMDP-based sleep scheduling algorithm with online planning (Dec-POP-SSA), to find suboptimal solutions of the sleep scheduling problem. This algorithm is executed in parallel by all the sensors in the same cluster, and is divided into three sub-phases: a planning phase, an executing phase

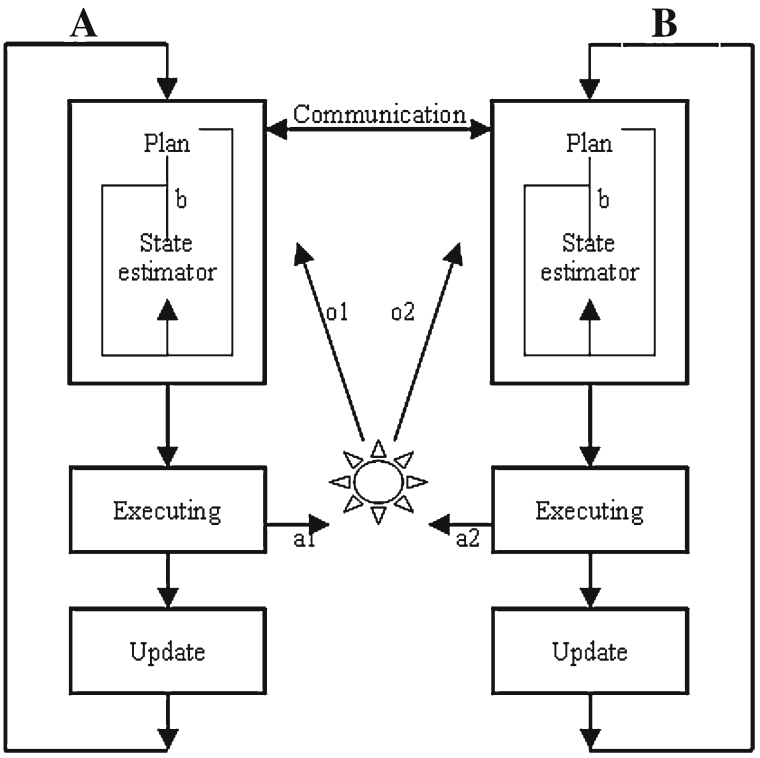

Fig. 2 The framework of Dec-POP-SSA

and an updating phase. In the planning phase, each sensor computes a joint policy for all possible histories in the belief pool, sensors in each cluster will be initialized with an empty history information and their sub-belief pool as b^0. During the executing phase, each sensor adds its new local observation to its own local history, and executes an action according to its component in the joint policy and its current local history. After that, during the updating phase, each sensor updates its own sub-belief pool based on the plan. An example involving two sensors that are in the same cluster is illustrated in Fig. 2. Two main problems need to be solved in the design:

1. how to maintenance the belief states;
2. how to solve the joint policy.

4.1 Phase 1: generate the belief pool

For the first problem, an efficient data structure must be designed to represent a belief pool and the sensors' historical information. The sub-belief pool stores the sub-belief state with every clusters. When storing the history, it does not need to save the whole action observation sequence in the belief pool, and it does not need the joint history information of all the sensors, it only needs to maintain the sub-

belief pool, which includes sub-joint history information of sensors in each cluster (x). A hash table $h^t_{(x)} = \langle \theta_{(x)} b^t_{(x)} \rangle$ is used to represent the joint history, where $\overrightarrow{\theta_{(x)}} = \langle \theta_{(1)}, \theta_{(2)}, \ldots, \theta_{(|C_{(x)}|)} \rangle$ is the joint index to a history, $|C_{(x)}|$ represents the number of nodes in the cluster (x), $\theta_{(i)}$ is the history index for sensor i, $b^t_{(x)}$ is the joint belief in the cluster, which is the joint history information induced within the cluster. That is, the state probability distribution of the sensors in the same cluster. In each sub-belief pool, since only one history is kept for one policy, the history index for sensor i is represented as a tuple $\theta_{(i)} = \langle a_i^{t-1}, o_i^t \rangle$ where a_i^{t-1} is the policy tree index of the previous step and o_i^t is the observation of the current step. Therefore, the data structure is

$$h^t_{(x)} = \langle \langle a_1^{t-1}, o_1^t \rangle, \langle a_2^{t-1}, o_2^t \rangle, \ldots, \langle a_n^{t-1}, o_n^t \rangle, b^t_{(x)} \rangle. \tag{13}$$

To compute a set of joint belief states of the current step, usually use Bayes' rule from the previous step. In this paper, since the state transition and observation function is unknown, it cannot compute a set of joint belief states through the Bayes' rule.

Beliefs generating is a typical nonlinear filtering problem, due to the influence of the weak system observability, initial value error and its covariance are very large, the standard particle filter algorithm is prone to degradation and rapid particle impoverishment phenomenon, filtering performance is poor. Monte Carlo random sampling cannot effectively simulate data completely, it will influence the estimation precision, since the sample swatches will be gathered and gap phenomenon, can't fully get the sample space.

One major advancement has come in the development of alternatives to the pseudo-random or pseudo-Monte Carlo (PMC) number sequences that the simulation processes are based on [34]. The use of so-called quasi-random number sequences or quasi-Monte Carlo sequences (QMC) can lead to significant savings in simulation and estimation processes, by enabling stable performance with a lower number of draws. In transportation research, the most commonly used type of quasi-random number sequences is the Halton sequence [35,36].

Definition 5 (*Halton sequence*) [36] For any prime number q and any positive integer D, there is a unique base q representation

$$D = \sum_{j \geq 0} Z_j q^j. \tag{14}$$

The $Z_j = Z_j(q, D) \in \{0, \ldots, q-1\}$ are the 'bits' of D in base q. For any $D < q^m$, this sequence has all zero entries in positions $j \geq m$. With the bit sequence $(Z_j) = (Z_j(D))$ in hand, it is defined as

$$\beta_q(D) := \sum_{j \geq 0} Z_j q^{-j-1}. \tag{15}$$

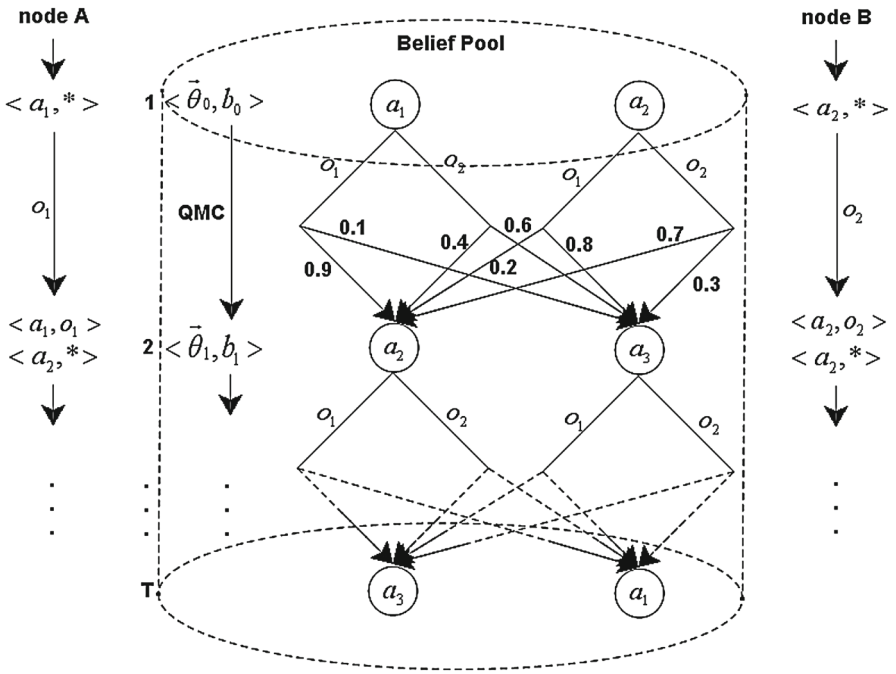

Fig. 3 The example of the one sub-belief pool

Given the space dimension $d \geq 1$, the first d prime numbers $y_1, \ldots, y_d$ are chosen. The sequence of points $(\widehat{x}_k)_{k \in N}$ in $[0, 1]^d$ is then defined by

$$\widehat{x}_k := (\beta_{y_1}(k), \ldots, \beta_{y_d}(k)). \tag{16}$$

The quasi-Monte Carlo method will be used to compute the joint belief states. For each decision cycle t, select $\mathcal{K}$ deterministic sample points $\left\{x_t^{(i)}\right\}_{i=1}^{\mathcal{K}}$ by QMC method, which can get the estimation precision better than that of Monte Carlo Method. Then the simulator can be utilized to calculate state distribution in the absence of a complete model. But the amount of calculation growth index form. Carrier Aggregation is used for all clusters, to reduce the sampling size at the same time to ensure that samples move to the high likelihood, to improve the sampling accuracy.

At the end of each decision cycle, the information index is updated for each of the historical. Then, the cluster head in each cluster, sample joint history by QMC method, and calculates the corresponding joint beliefs in the cluster. Historical information is a sequence, which contains a sensor's action on each step and the observation from the cluster environment. The example of one sub-belief pool is shown in Fig. 3, in the belief pool, for the node whose action is a_1, when it obtains the observation o_1,the probability of transition action to a_2 is 0.9, and the probability of transition action to a_3 is 0.1. When this node obtains the observation o_2, the probability of transition action to a_2 is 0.4, and the probability of transition action to a_3 is 0.6. The belief states are

obtained by the QMC method. Each node can decide its next action, depending upon this belief pool and the node's current action. For example, for the node A, assume its current action is a_1, if it obtains the local observation o_1, according to the belief pool, it will transition action to a_2. For the node B, assume its current action is a_2, if it obtains the local observation o_2, according to the belief pool, its action is not changed. The algorithm of generating the state distribution by QMC method is shown in Algorithm 1.

Algorithm 1: generate the state distribution-QMC
step 1: the state distribution at time $k-1$ is $P(x_{k-1}) = N(x_{k-1}; \overline{x}_{k-1}; \widehat{P}_{k-1})$
step 2: QMC sampling: Generate a length of $\mathcal{K}$ points that is obey the distribution of $P(x_{k-1})$
step 3: prediction a collection $\{x_{k|k-1}^{(i)}\}_{i=1}^{\mathcal{K}}$ of k moment, estimate the variance and covariance:

$$\overline{x}_{k|k-1} = \frac{1}{\mathcal{K}} \sum_{i=1}^{\mathcal{K}} x_{k-1}^{(i)}, \widehat{P}_{k|k-1} = \frac{1}{\mathcal{K}} \sum_{i=1}^{\mathcal{K}} (x_{k-1}^{(i)} - \overline{x}_{k|k-1})(x_{k-1}^{(i)} - \overline{x}_{k|k-1})^{\mathcal{T}}.$$

step 4: use the observed value o_k to calculate the weights of each sample $\omega_k^{(i)}$, then normalize processing further:

$$\omega_k^{(i)} = P(o_k|x_k^{(i)}) N(x_k^{(i)}; \overline{x}_{k|k-1}; \widehat{P}_{k|k-1}) / q(x_k|o_{1:k}), \overline{\omega_k^{(i)}} = \omega_{\mathcal{K}}^{(i)} / \sum_{i=1}^{K} \omega_k^{(i)}.$$

step 5: estimate for target state and posterior distribution at time k is

$$\overline{x}_k = \sum_{i=1}^{\mathcal{K}} \omega_k^{(i)} x_k^{(i)}, \widehat{P}_k = \sum_{i=1}^{\mathcal{K}} \omega_k^{(i)} (x_k^{(i)} - \overline{x}_k)(x_k^{(i)} - \overline{x}_k)^{\mathcal{T}}.$$

4.2 Phase 2: history merging

During the executing phase, note that the observations of other sensors in the same cluster are not available. Each sensor must consider about all the possible histories that could be observed by the other sensors in the same cluster, and it need to know how that may affect its own action selection. However, the number of possible joint histories increases exponentially with the horizon. A more detailed analysis of the planning phase shows that most of the joint histories kept in memory are useless. While each sensor does not share private information with each other in the whole network, the sensors in a cluster can maintain the histories based on the information they have.

By the online planning, the policy equivalent condition facilitates the sub-policy reuse, which is to map several histories to a single sub-policy. It is also called merging when handling several histories. For example, some active sensors do not observe events after a period of time, they go to the sleep mode, there is no meaning of the state history before. If there are more than two similar clusters environments, the same policy can be taken among them, then we choose any one history to be recorded in the belief pool. Furthermore, in the same cluster, it requires to know h_{-i}, every possible historical combination of other sensors. Since every single value may affect the policy of sensor i, considering every possible h_{-i} is important. Algorithm 2 describe the history merging algorithm in detail.

Algorithm 2: history merging
(for each node i in the cluster x)
step 1: $\widetilde{H}_i^t \leftarrow \phi$,$\overline{H}_i(\mu_i) \leftarrow \phi$
step 2: for each $h_i \in H_i^t$ do
  step 2.1: Search the optimal policy μ_j in the belief pool
  step 2.2: Obtain the optimal policy μ_i according h_i
  step 2.3: if $R(\mu_i) > R(\mu_j) + \varepsilon_1$
    step 2.3.1: $\overline{H}_i(\mu_i) \leftarrow \overline{H}_i(\mu_i) \cup \{h_i\}$
step 3: for each μ_i do
  step 3.1: if $\overline{H}_i(\mu_i)$ is not empty then
    step 3.1.1: randomly selected a h_i from $\overline{H}_i(\mu_i)$
    step 3.1.2:$\widetilde{H}_i^t \leftarrow \widetilde{H}_i^t \cup \{h_i\}$
step 4: return $\widetilde{H}_i^t$ to the sub-belief pool of cluster (x)

For the second problem: how to solve the joint policy, without knowledge of the observations of the other sensors, every sensors must consider about all the possible belief states that could be held by others. In order to find sensor i's sleep scheduling policy μ_i for history h_i, i need to consider about every possible histories h_{-i} in the same cluster, the possible policies associated with them are also to considered. That is to say, one joint policy $\overrightarrow{\mu}$ need to be found, which can maximise the reward function as follows:

$$R(\overrightarrow{\mu}) = \sum_{h \in H} \sum_{s \in S} pR(\mu(h), s). \tag{17}$$

The goal of the Dec-POP-SSA is that each sensor independently calculates the joint policy $\mu(h)$ for the cluster and then executes its share of the policy based on its own local history.

As Dec-POP-SSA algorithm consists of three phases, this section describes these three phases in detail. In the planning phase, each node computes its own policy by offline, the policy represented as a layered structure, a layer of policy represents a decision cycle. A fixed number of decision nodes are at each layer, and each node contains the selected actions and belief distribution calculated by QMC sampling. In the executing phases, according to the action distribution, each node chooses an action, then according to the observation from the environment, each node moves to the next layer. In the online policy, the execution of the algorithm is simple and clear. So the key is the way to generate such a policy, which is created in the planning stage. In every iteration step, a policy is chosen for each node, and then calculate state distribution through QMC sampling. In the updating phase, according to the distribution, optimize and ameliorate the policy, the process continues, until there are no policies for further improvement.

As shown in Fig. 2, it can complete information sharing between the sensors before entering the planning phase. But in the GWSNs, communication is costly if the active sensors are far from each other, the frequency of communication should be limited for each sensor. In the follow possible way, sensors can share information and coordinate their actions, the belief pool was detected not agree with the information from the

environment. That is to say belief inconsistency is detected when the sensor's local observation does not agree with the projected joint belief.

Definition 6 (*the observation inconsistency*) For the sensor i, at time slot t, the sub-belief pool B^t that is obtained by QMC is said to be ε_2-inconsistency with i's observation o_i^t if

$$|b(s) - p_{(x)}^t(b)| < \varepsilon_2, \tag{18}$$

where $p_{(x)}^t(b)$ is the sub-belief in cluster (x) at time slot t by using QMC sampling, sensor i is in the cluster (x).

This definition is used to monitor inconsistency between sensor i's local history and observation, which provides an indication of historical inconsistency in the pool. If the inconsistent is occurred, the sub-belief pool can be refreshed by communicating with the other sensors, then the sub-belief pool contains the real joint history, and it is consistent with the observation, which leads to gain better joint policy, at the same time also reflect the value of the communication for decision-making.

There is an issue that how to choose the heuristic policy, the better calculation result can be gained by combining different heuristic policy. In the model-free environment, the most simple heuristic policy is random policy, that is each sensor chooses an action randomly. Another one, it can use manual coding policy according to common sense. The performance of the online algorithm depends on the heuristic algorithm, the stand or fall of the heuristic algorithm mainly related to the actual problem. In this paper, based on the structure of the cluster, let the cluster-heads action and let the cluster members go to sleep is a best heuristic algorithm. After the heuristic algorithm is given, the Dec-POP-SSA algorithm is to obtain the approximate optimal combination policy by constructing and resolving a series of linear programming.

To start the algorithm, each local policy is initialized to be deterministic for each sensor, by selecting the man-made manual coding policy. Then, each sensor i's policy is improved, which is done for i by finding the best policy $\mu_i(q_i|h_i)$ as satisfying

$$R(\overrightarrow{\mu}) + \varepsilon_1 \leq \sum_{i \in C_{(x)}} p_{(x)}^k(b) \left[\sum_{\overrightarrow{a}} \mu_i(a_i|h_i)\mu_{-i}(a_{-i}|h_{-i})R(\overrightarrow{a}, b(h)) \right], \tag{19}$$

where $\mu_{-i}(a_{-i}|h_{-i}) = \prod_{k \neq i} \mu_k(a_k|h_k)$.
Algorithm 3 describe the Dec-POP-SSA in detail:

5 Theoretical analysis

Deriving an upper bound of the expected reward is by a similar method of [18,20]. This section derives an upper bound for the state space with QMC sampling. First, Lemma 1 provides an upper bound on the expected reward by one sensor; second, Lemma 2 provides an upper bound on the expected reward by one cluster; finally,

Theorem 2 provides an upper bound on the expected reward for the whole network, and Theorem 3 provides the space and time complexity of Dec-POP-SSA.

Algorithm 3: Dec-POP-SSA
step 1:generate the initial belief, sampling $x_0^{(i)} \sim p(x_0)$,
step 2: call algorithm 1, update the new belief distribution;
step 3: for each $i_{(x)} \in C_{(x)}$ do
step 3.1: manual heuristic policy $\overrightarrow{a}^0$, where the cluster-head $r_{CH} = 0$, and the cluster-member $r_{CM} = r_0$;
step 3.2: calculate the initial rewards $R_x^0(p_o, r_o)$;
step 3.3: $comm \leftarrow 0; H^0 \leftarrow \{\overrightarrow{a}^0\}$; $t = 1$;
step 3.4: while($t \leq \mathcal{T}$)
step 3.4.1: call algorithm 1, update b^t;
step 3.4.2: $H^t, b^t, o_i^t, h_i^t \leftarrow$ the observation from the environment, update node i's own local history with o_i^t and calculative policy;
step 3.4.3: if (H^t and o_i^t are inconformity) $comm \leftarrow 1$, construct h_i^t, call algorithm 1, update b^t, select an action $\overrightarrow{a}^t$, based on the maximum rewards $R_x^t(p^t, r_t)$; $H^t \leftarrow \{h^t\}, b^t \leftarrow \{b^t\{h^t\}\}$
step 3.4.4: else select an action $\overrightarrow{a}^t$ for H^t, b^t ; update node i's own local history h_i^t, according to $\overrightarrow{a}^t$;
step 3.5: t++;
step 4: for each cluster do
step 4.1: generate $|C|$ aggregate particles $\{(x_k^{ti}, \omega_k^{ti})\}_{i=1}^{|C|}$,
step 4.2: aggregation the particles

$$R^* = \sum_{i=1}^{|C|} R_{(i)}^* \omega_k^i / \sum_{i=1}^{|C|} \omega_k^i$$

Lemma 1 *Given the current belief p^k which is obtained from the QMC sampling, and an action vector u_k which is computed through the Dec-POP-SSA algorithm, the current residual sleep times vector r_k, the coverage intensity as well as connectivity intensity, the expected reward is upper bounded by*

$$E[R \mid p^k, u_k] \leq E(R_0) - \gamma_2 \sum_{i=1}^{m} p_i^k \sum_{j=1}^{m} p(d_{k+1} = j \mid d_k = i) \times \max_{k \neq j} Q\left(\frac{d_{kj}}{2} + \frac{ln\frac{[p^{k+1}]_j}{[p^{k+1}]_k}}{d_{kj}}\right). \tag{20}$$

Proof An upper bound on the reward is derived, by giving the current belief p^k which is obtained from the QMC sampling. Those active sensors must meet certain connectivity and coverage intensity. When the distribution of the state is obtained according to the QMC sampling, and the requirements of basic connectivity and coverage can be met, this is the upper bound of the reward. A large number of literatures are studied on network connectivity and coverage, they found that the number of active sensor related to the network area. When the network area is fixed. Then the number of active sensors is fixed, specific value is determined by the actual. This article did not describe detail, the basic reward is defined in advance:

$$E(R_0) = E\left[\sum p^k(b)\left(\sum_{l=1}^{n} \mathbb{1}_{\{r_l^k=0\}} \gamma_1(\text{Cov} + \text{Con}) - \gamma_2 \sum_{l=1}^{n} c\mathbb{1}_{\{r_l^k=0\}}\right)\right]. \tag{21}$$

$TR = \mathbb{1}_{\{r_k=0\}} e_k - \mathbb{1}_{\{r_k>0\}} o_k$ is the tracking error, the expected reward can be written as

$$E\left[R \mid p^k, u_k\right] = E(R_0) - E[\gamma_2(TR)], \tag{22}$$

where

$$\begin{aligned} E[\gamma_2(TR)] &= E\left[\gamma_2\left(\mathbb{1}_{\{r_k=0\}} e_k - \mathbb{1}_{\{r_k>0\}} o_k\right)\right] \\ &= E\left[\gamma_2\left(\sum_{j=1}^{m} Pr\left[\widehat{d}_{k+1} \neq j \mid p^k, u_k, r_k, d_{k+1} = j\right] \times Pr\left[d_{k+1} = j \mid p^k, u_k\right]\right)\right] \\ &= \gamma_2\left(\sum_{i=1}^{m} p_i^k \sum_{j=1}^{m} p\left(d_{k+1} = j \mid d_k = i\right) \times Pr\left[\widehat{d}_{k+1} \neq j \mid p^k, u_k, r_k, d_{k+1} = j\right]\right). \end{aligned} \tag{23}$$

If the states obtained by QMC are almost close to the real distribution, the reward is the upper bound. The real observation distribution is defined as:

$$P(E \mid H_j) = Pr[\widehat{d}_{k+1} \neq j \mid p^k, u_k, r_k, d_{k+1} = j]. \tag{24}$$

Using standard analysis for likelihood ratio tests [20,38], we have

$$P(E \mid H_j) \geq \max_{k \neq j} Q\left(\frac{d_{kj}}{2} + \frac{ln\frac{[p^{k+1}]_j}{[p^{k+1}]_k}}{d_{kj}}\right), \tag{25}$$

where $d_{kj}^2 = \frac{\Delta m_{kj}^T \Delta m_{kj}}{\sigma^2}$, $\Delta m_{kj} = m_k - m_j$, m_j is the mean received signal strength when the target is at state j. $Q(.)$ is the normal distribution, the quantity d_{kj} plays the role of distance between the two hypothesis and hence depends on the difference of their corresponding mean vectors and the noise variance σ^2.

The network involves a communication channel that is assumed perfectly reliable, it is assumed to be always available and without noise so that $\sigma = 1$. The belief state at time slot k in cluster (x) by QMC sampling is defined as $p_{(x)}^k(b)$, the same to the initiation belief state $b^0 = p(b)$, it is also a $|(x)| \times |(x)|$ matrix vector. The jth element is $[p_{(x)}^k(b)]_j$, which entry equals to the belief state in cluster j. For simplicity, it is defined as $\left[p^k\right]_j = \left[p_{(x)}^k(b)\right]_j$ in this section.

So, we have

$$E\left[R \mid p^k, u_k\right] \leq E(R_0) - \gamma_2 \sum_{i=1}^{m} p_i^k \sum_{j=1}^{m} p(d_{k+1} = j \mid d_k = i) \times \max_{k \neq j} Q\left(\frac{d_{kj}}{2} + \frac{ln\frac{[p^{k+1}]_j}{[p^{k+1}]_k}}{d_{kj}}\right). \tag{26}$$

□

Lemma 2 *The expected reward by one cluster* (x) *is upper bounded by:*

$$E\left[R_{(x)} \mid p^k, u_k, r_k\right] \leq \sum_{l \in C_{(x)}} p^k(b) \left\{ \mathbb{1}_{\{r_l^{k+1}=0\}} \sum_{i=1}^{m} p_i^k E_0(p; i, l) + \mathbb{1}_{\{r_l^{k+1}>0\}} \sum_{i=1}^{m} p_i^k E_{0_{-l}}(p; i, l) \right\}. \tag{27}$$

Proof The effect of each sensor on the reward is:

$$E_0(p; i, l) = E(R_0) - \gamma_2 \sum_{j=1}^{m} p(d_{k+1} = j \mid d_k = i) \times \max_{k \neq j} Q_{\mathbb{1}_{\{r_l^{k+1}=0\}}}\left(\frac{d_{kj}(\mathbb{1})}{2} + \frac{ln\frac{[p^{k+1}]_j}{[p^{k+1}]_k}}{d_{kj}}\right), \tag{28}$$

which is the reward that all sensors will be active at the next time sole.

$$E_{0_{-l}}(p; i, l) = E(R_0) - \gamma_2 \sum_{j=1}^{m} p(d_{k+1} = j \mid d_k = i) \times \max_{k \neq j} Q_{\mathbb{1}_{-l}}\left(\frac{d_{kj}(\mathbb{1}_{-l})}{2} + \frac{ln\frac{[p^{k+1}]_j}{[p^{k+1}]_k}}{d_{kj}}\right). \tag{29}$$

Let $\mathbb{1}_{-l}$ denote a vector of length n with all entries equal to one except for the l-th entry being zero.

The expected reward can be readily upper bound in cluster (x):

$$E[R_x \mid p^k, u_k, r_k] \leq \mathbb{1}_{\{r_l^{k+1}=0\}} E[R_x \mid p^k, r_{k+1} = 0] + \mathbb{1}_{\{r_l^{k+1}>0\}} E[R_x \mid p^k, r_i^{k+1} = 0, \quad \forall i \neq l]. \tag{30}$$

Since this holds for every $l \in C_{(x)}$, an upper bound on the expected reward in cluster (x) can be written as a convex combination of all sensors contributions

$$
\begin{aligned}
&E[R_x \mid p^k, u_k, r_k] \\
&\leq \sum_{l \in C_{(x)}} p^k(b)\{\mathbb{1}_{\{r_l^{k+1}=0\}} E[R_x \mid p^k, r_{k+1}=0] + \mathbb{1}_{\{r_l^{k+1}>0\}} E[R_x \mid p^k, r_l^{k+1} \\
&= 0, \forall i \neq l]\} \\
&\leq \sum_{l \in C_{(x)}} p_k(b) \left\{ \mathbb{1}_{\{r_l^{k+1}=0\}} \left(E(R_0) - \gamma_2 \sum_{i=1}^{m} p_i^k \sum_{j=1}^{m} p(d_{k+1}=j \mid d_k=i) \right.\right. \\
&\quad \left. \times \max_{k \neq j} Q_{\mathbb{1}_{\{r_l^{k+1}=0\}}} \left(\frac{d_{kj}(\mathbb{1})}{2} + \frac{ln\frac{[p^{k+1}]_j}{[p^{k+1}]_k}}{d_{kj}} \right) \right) + \mathbb{1}_{\{r_l^{k+1}>0\}} (E(R_0) \\
&\quad -\gamma_2 \sum_{i=1}^{m} p_i^k \sum_{j=1}^{m} p(d_{k+1}=j \mid d_k=i) \\
&\quad \left. \times \max_{k \neq j} Q_{\mathbb{1}_{-l}} \left(\frac{d_{kj}(\mathbb{1}_{-l})}{2} + \frac{ln\frac{[p^{k+1}]_j}{[p^{k+1}]_k}}{d_{kj}} \right) \right) \right\}. \qquad (31)
\end{aligned}
$$

So

$$
\begin{aligned}
E[R_x \mid p^k, u_k, r_k] \leq \sum_{l \in C_{(x)}} p^k(b) \Bigg\{ & \mathbb{1}_{\{r_l^{k+1}=0\}} \sum_{i=1}^{m} p_i^k E_0(p; i, l) \\
& + \mathbb{1}_{\{r_l^{k+1}>0\}} \sum_{i=1}^{m} p_i^k E_{0_{-l}}(p; i, l) \Bigg\}. \qquad (32)
\end{aligned}
$$

□

Theorem 2 *An upper bound on the Dec-POP-SSA algorithm at belief state b^0 can be obtained as a solution to the following optimization problem with QMC sampling:*

$$
\begin{aligned}
R^* = max \Bigg\{ & \sum_{l=1}^{|C|} \max_{u_l} \Bigg\{ \sum_{j=0}^{u-1} \sum_{i=1}^{|C_{(i)}|} [p_{(i)}^j(b)]_i E_{0_{-l}}(i,l) + \sum_{i=1}^{|C_{(i)}|} [p_{(i)}^u(b)]_i E_0(i,l) \\
& - c \sum_{i=1}^{C_{(x)}} \left[b_j p_{(i)}^{u+1}(b) \right]_i + \sum_{i=1}^{|C_{(i)}|} \left[p_{(i)}^{u+1}(b) \right]_i J^l(b_i) \Bigg\} \omega_k^l / \sum_{l=1}^{|C|} \omega_k^l \Bigg\}. \qquad (33)
\end{aligned}
$$

Proof If the QMC sampling can be perfectly observed after taking the sleeping action, an optimal policy similar to [17] could be found by solving the Bellman equation, which is an equation for the per-sensor problem under the assumption of no future observations. Note that it considers two cases. The first one is for a residual sleep time

of 0, the second one is for a residual sleep time great than 0. In each case, the two terms inside the maximization represent the basic reward and the tracking cost. b_i equals to the initiation belief state in cluster (i), and there are $|C_{(i)}|$ sensors in the cluster (i). So, we have

$$R^*_{(l)} = \max_{l \in C_{(i)}} \left\{ \mathbb{1}\{r_l^k = 0\} \times \left(\sum_b p(b) E_0(p; b, l) + \sum_{i=1}^{|C_{(i)}|} [p^1_{(i)}(b)]_i R(b_i, 0) \right) + \mathbb{1}\{r_l^k > 0\} \times \left(\sum_b p(b) \lambda_l E_{0_{-l}}(p; b, l) + \sum_{i=1}^{|C_{(i)}|} [p^1_{(i)}(b)]_i R(b_i, u_l) \right) \right\}. \quad (34)$$

$R(b_i, u)$ can be obtained by recursive iteration until the system reaches $(b_i, 0)$,

$$R_l(b_j, u) = \lambda_l^j E_{0_{-l}}(b_0; i, l) + \sum_{i=1}^{|C_{(i)}|} [b_j p_{(i)}(b)]_i R^l(b_i, u-1), \quad (35)$$

and

$$R_l(b_j, 1) = \lambda_l^j E_0(b_0; i, l) - c \sum_{i=1}^{|C_{(i)}|} [b_j p_{(i)}(b)]_i + \sum_{i=1}^{|C_{(i)}|} [b_j p_{(i)}(b)]_i J^l(b_i, 0). \quad (36)$$

Assume u_l is a controllable behavior, which is the behavior of the sensor l under a given belief state $p(b)$. Since an action needs to be made when the sensor wakes up, the actions at each $(b_i, 0)$ need to be defined. It is easy to see that an upper bound cluster (i) can be obtained as a solution of the following maximization problem as Eq. (37).

$$R^*_{(l)} = \max_{u_l} \left\{ \sum_{j=0}^{u-1} \sum_{i=1}^{|C_{(i)}|} \left[p^j_{(i)}(b) \right]_i E_{0_{-l}}(i, l) + \sum_{i=1}^{|C_{(i)}|} \left[p^u_{(i)}(b) \right]_i E_0(i, l) - c \sum_{i=1}^{C_{(i)}} \left[b_j p^{u+1}_{(i)}(b) \right]_i + \sum_{i=1}^{|C_{(i)}|} \left[p^{u+1}_{(i)}(b) \right]_i J^l(b_i) \right\}. \quad (37)$$

According to aggregation particles of the Dec-POP-SSA algorithm, we assume that there are $|C|$ clusters in the networks. Then $R^* = \sum_{l=1}^{|C|} R^*_{(l)} \omega_k^l / \sum_{l=1}^{|C|} \omega_k^i$.

Together with Eq. (37), we can determine an upper bound on the total expected reward,

$$R^* = \max \left\{ \sum_{l=1}^{|C|} \max_{u_l} \left\{ \sum_{j=0}^{u-1} \sum_{i=1}^{|C_{(i)}|} \left[p_{(i)}^{j}(b) \right]_i E_{0_{-l}}(i,l) + \sum_{i=1}^{|C_{(i)}|} \left[p_{(i)}^{u}(b) \right]_i E_0(i,l) \right.\right.$$
$$\left.\left. -c \sum_{i=1}^{C_{(i)}} \left[b_j p_{(i)}^{u+1}(b) \right]_i + \sum_{i=1}^{|C_{(i)}|} \left[p_{(i)}^{u+1}(b) \right]_i J^l(b_i) \right\} \omega_k^l / \sum_{l=1}^{|C|} \omega_k^l \right\}. \tag{38}$$

□

Theorem 3 *The space complexity of Dec-POP-SSA is $\mathcal{O}(n\mathcal{T}\mathcal{K})$, time complexity is $\mathcal{O}(\mathcal{T}^2)$, where n is the number of the nodes in the network, $\mathcal{T}$ is the number of the policy steps, and $\mathcal{K}$ is the minimum number of sampling.*

Proof First, compute the minimum number of samples. Assume that $\mathcal{X}$ is the random variable which is in $[\mathcal{X}_{\min},\mathcal{X}_{\max}]$, and $\overline{\mathcal{X}} = E[\mathcal{X}]$ is $\mathcal{X}$'s expectation depending upon the QMC method. Suppose that $\{x_1, x_2, \ldots, x_{\mathcal{K}}\}$ is a sample sequence, whose value is also in $[\mathcal{X}_{\min},\mathcal{X}_{\max}]$, and the mean value is $\widetilde{\mathcal{X}} = \frac{1}{\mathcal{K}} \sum_{k=1}^{\mathcal{K}} x_k$. In order to keep the samples mean $\widetilde{\mathcal{X}}$ close to the expectations $\overline{\mathcal{X}}$, enough samples need to be collected. However, the more samples taken, the higher the communication complexity is. We hope to collect less enough samples, but it can make the average close to expectations. Hoeffding inequality [38] is a very good method, which can be used to calculate the less number of samples. According to the Hoeffding inequality, we have

$$P(\widetilde{\mathcal{X}} \leq \overline{\mathcal{X}} + \delta) \geq 1 - e^{-2\mathcal{K}\delta^2/(\mathcal{X}_{\max}-\mathcal{X}_{\min})^2}, \tag{39}$$

and

$$P(\widetilde{\mathcal{X}} \geq \overline{\mathcal{X}} + \delta) \geq 1 - e^{-2\mathcal{K}\delta^2/(\mathcal{X}_{\max}-\mathcal{X}_{\min})^2}, \tag{40}$$

where δ refers to a smaller threshold.

Set another confidence threshold ϑ. If $\widetilde{\mathcal{X}}$ wants to meet the condition, the required minimum number of sampling is,

$$\mathcal{K} = \frac{(\mathcal{X}_{\max} - \mathcal{X}_{\min})^2 ln(1/\vartheta)}{2\delta^2}. \tag{41}$$

In Dec-POP-SSA algorithm, the number of the policy steps $\mathcal{T}$ is fixed, select a sensor is needed to improve the pre-generated policy. So for the cluster (x), which has $|C_{(x)}|$ sensors, the overall space requirements for storage is $\mathcal{O}(|C_{(x)}|\mathcal{T}\mathcal{K})$. Since there are n nodes in the network, the space complexity of Dec-POP-SSA is $\mathcal{O}(n\mathcal{T}\mathcal{K})$.

In the process of iteration, they need QMC samples for many times, the total time needed is $1 + 2 + \cdots + \mathcal{T}$, that is to say the decision steps are a quadratic function $\mathcal{O}(\mathcal{T}^2)$. □

6 Numerical analysis

This section gives some simulation results that illustrate the performance of the algorithm Dec-POP-SSA:

1. The QMC sampling can accurately describe the distribution of the state, which can be used for the Dec-POP-SSA to solve the sleep scheduling problem in the GWSNs.
2. The tracking errors by Dec-POP-SSA are lower than QMDP and FCR.
3. The Dec-POP-SSA algorithm has reached the highest reward.

It is assumed that events happen in some location with a multi-peak Gaussian distribution, which is structured as the Formula (42), the results of simulation runs were averaged to compute.

$$P(x, y) = \frac{(1-x)^2}{4} e^{-x^2-(y+1)^2} - \frac{5}{6}\left(\frac{x}{5} - x^3 - y^5\right) e^{-x^2-y^2} - \frac{1}{36} e^{-(x+1)^2-y^2} + \frac{1}{2}. \tag{42}$$

For the nonlinear filtering operation, the value of $p^k_{(x)}(b)$ for each cluster (x) and time slot k is generated by averaging 50 QMC simulations. The probability distribution of events happened in the designated area is shown in Fig. 4, and one example of the QMC simulations is also shown in this figure as '+' marked, which has 100 samplings one time.

In Fig. 5, there are samplings at 10 time slots, 100 QMC points are sampling each time slot, and the results of 50 simulations runs were averaged when plotting the points. From the figure, it is very clear that the value of $p^k_{(x)}(b)$ at every simulation is almost around 0.5, the peaks almost can be got. It is very consistent with multi-peak Gaussian distribution, which is defined by the Formula (42).

In the case of QMDP policies, the value of $p^k_{(x)}(b)$ is generated by 100 MC samplings, for detailed instructions, the distribution of a comparative analysis, Fig. 6 extracts one example of QMC simulation and MC simulation. The plot shows the result of regular Monte Carlo sampling, that is, 100 points selected randomly. Random points tend to form a cluster, over-sampling the unit square in some place, this lead to gaps in other places, where the sample space is not explored at all. It is therefore difficult to achieve the peak value when using the MC sampling. For example, for those 65–100 samplings, almost all of the values were at about 0.5, because of the samplings are too dense, it is difficult to show the distribution of the original belief. The plot shows the result of QMC, that is, 100 points of a two-dimensional Halton sequence. It can be observed that they can reach the peaks under the QMC sampling method. Its distribution on the basis of a very significant multi-peak Gaussian distribution. So, it is visible that the QMC sampling can accurately describe the distribution of the state, which can be used for the Dec-POP-SSA to solve the sleep scheduling problem in the GWSNs.

The performance of the Dec-POP-SSA is illustrated in numerical analysis, compared to the QMDP versions of the greedy algorithm and the FCR versions of greedy

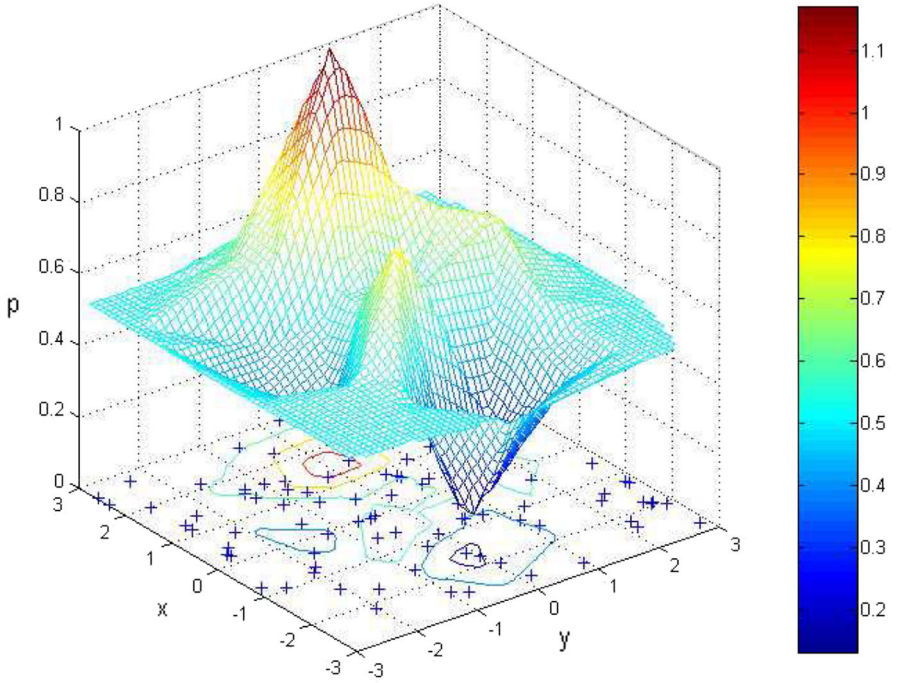

Fig. 4 The probability distribution of events and the example of the QMC

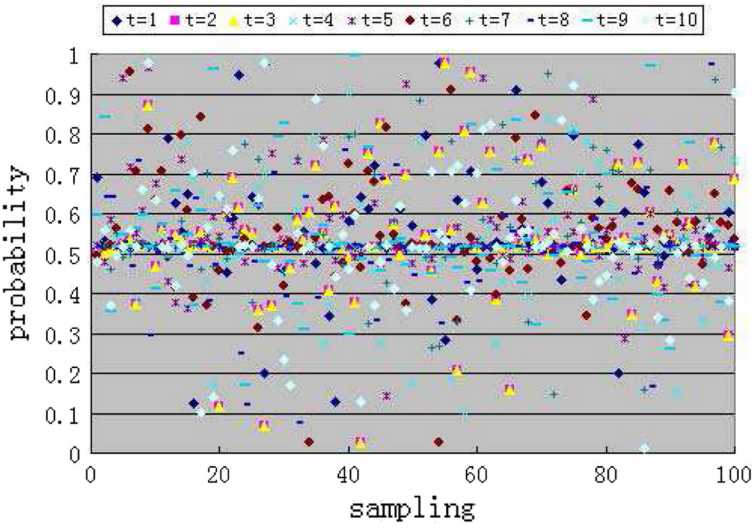

Fig. 5 The sampling at 10 time steps

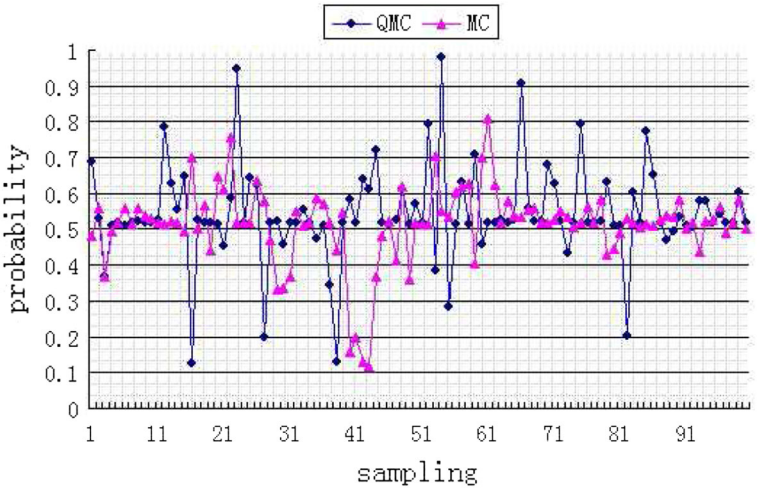

Fig. 6 The QMC and MC sampling at one time

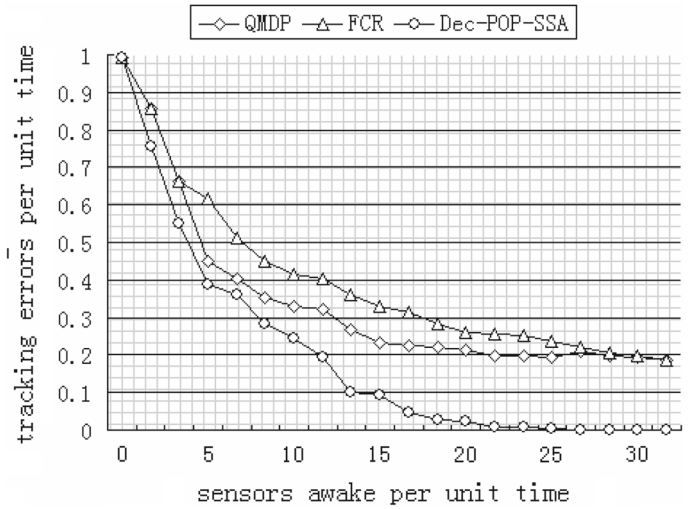

Fig. 7 The tracking errors by QMDP, FCR and Dec-POP-SSA

algorithm [19]. Analogous to [19], the algorithms are executed in the network, where the sensor field is sufficiently dense so that the sensors cover the entire area of interest, that is to say, the algorithms can satisfy the basic connectivity and coverage intensity. The 100 sensors with transmission radius 20 are randomly generated in the 100×100 m^2 network simulation area. The results of 50 simulation runs were averaged, as shown in Fig. 7, the x-axis is the percent of the wake sensors per unit time, the y-axis is the percent of the tracking error. The following conclusions can be drawn from it, the lower bound due to the Dec-POP-SSA assumption is tight when only a few sensors are awake; this is because the Dec-POP-SSA assumption incorporates only observa-

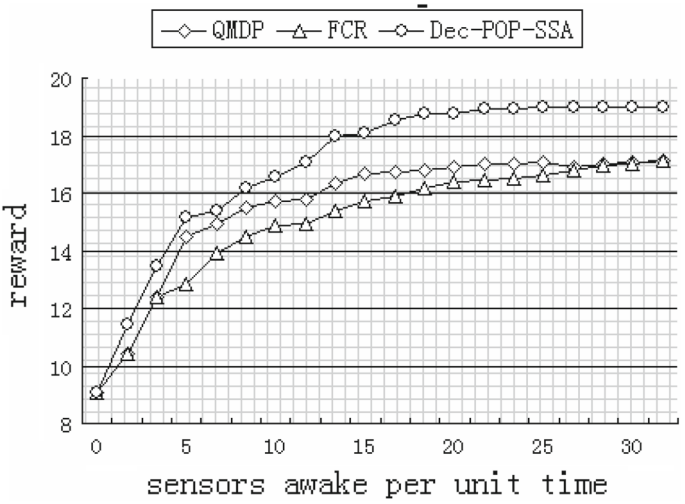

Fig. 8 The rewards by QMDP, FCR and Dec-POP-SSA

tion errors and when few sensors are making observations. The FCR policy is the worst-performing policy, since it uses the fixed belief while not real-time. The QMDP policy uses the MC, but it is too dense, so it is near the FCR when the awake nodes increase. On the whole, the tracking errors by Dec-POP-SSA are lower than QMDP and FCR.

The basic reward is set as $R_0 = 20$, when achieve the basic connectivity and coverage, the thresholds for Dec-POP-SSA are set as $\varepsilon_1 = \varepsilon_2 = \delta = 0.1$, and the discount factor are set as $\gamma_1 = \gamma_2 = 0.5$ in these experiments. The reward curves are shown in Fig. 8, the Dec-POP-SSA algorithm has reached the highest reward. The key reason lies in that the beliefs distribution is real-time, and adopts the way of collaboration, so as to achieve the approximate optimal benefits. However, other algorithms based on the assumption of a constant a belief distribution, in this hypothetical case, do not take into consideration the real-time and collaborative.

7 Conclusion

Sleep scheduling is one of the efficient strategies to achieve energy savings in GWSNs. In this paper, we proposed a novel scheme for the sleep scheduling based on Dec-POMDP. Then a sleep scheduling algorithm with online planning (Dec-POP-SSA) with respect to Dec-POMDP is presented. We give the theoretical analysis on the upper bound for Dec-POP-SSA. The numerical experiments' results illustrate that Dec-POP-SSA receives the highest reward over a multi-peak Gaussian distribution when compared with other approaches.

There are several avenues for energy saving in GWSNs such as sleep scheduling and network coding, both of which are two major techniques. Compared with sleep scheduling which saves energy by turning some redundant sensors into sleep mode,

network coding increases energy efficiency and reduces network congestion by combining packets destined for distinct users. How to reconcile these two techniques so that energy saving is achieved even more efficiently is worthwhile to study. Second, we can examine the sleep scheduling problem under the sophisticated attacker in unsafe network.

Acknowledgments The authors wish to thank National Natural Science Foundation of China (Grant No: 61072080, No. U1405255). Fujian Normal University Innovative Research Team (No. IRTL1207). The Natural Science Foundation of Fujian Province (No: 2013J01222, J01223, 2013J01221). The Education Department of Fujian Province science and technology project (JA13215).

References

1. Wei Q, Jianfeng G, Changqiao X, Hongke Z (2012) G-MER: green mobility estimation-based routing protocol in wireless sensor network. China Commun 9(5):10–21
2. Morreale P, Qi F, Croft P (2011) A green wireless sensor network for environmental monitoring and risk identification. Int J Sens Netw 10(1):73–82
3. Bianzino AP, Chaudet C, Rossi D, Rougier JL (2012) A survey of green networking research. IEEE Commun Surv Tutor 14(1):3–20
4. Chen Y, Zhang S, Xu S, Li GY (2011) Fundamental trade-offs on green wireless networks. IEEE Commun Mag 49(6):30–37
5. Jabbar S, Minhas AA, Paul A, Rho S (2014) Multilayer cluster designing algorithm for lifetime improvement of wireless sensor networks. J Supercomput 70(1):104–132
6. Kim YH, Han YH, Jeong YS, Park DS (2013) Lifetime maximization considering target coverage and connectivity in directional image/video sensor networks. J Supercomput 65(1):365–382
7. Wang WN, Mi ZK, Wang B (2012) Green networking based energy efficient communications. China Commun 9(2):22–30
8. Zeng Y, Sreenan CJ, Xiong N, Yang LT, Park JH (2010) Connectivity and coverage maintenance in wireless sensor networks. J Supercomput 52(1):23–46
9. Zhao Y, Wu J, Li F, Lu S (2012) On maximizing the lifetime of wireless sensor networks using virtual backbone scheduling. IEEE Trans Parallel Distrib Syst 23(8):1528–1535
10. Nan G, Shi G, Mao Z, Li M (2012) CDSWS: coverage-guaranteed distributed sleep/wake scheduling for wireless sensor networks. EURASIP J Wirel Commun Netw 1:1–14
11. Liu C, Wu K, Xiao Y, Sun B (2006) Random coverage with guaranteed connectivity: joint scheduling for wireless sensor networks. IEEE Trans Parallel Distrib Syst 17(6):562–575
12. Wang L, Wei RZ, Tian ZH (2012) Cluster based node scheduling method for wireless sensor networks. Sci China Inf Sci 55(4):755–764
13. Ding Y, Wang C, Xiao L (2009) An adaptive partitioning scheme for sleep scheduling and topology control in wireless sensor networks. IEEE Trans Parallel Distrib Syst 20(9):1352–1365
14. Yu J, Zhang Q, Yu D, Chen C, Wang G (2014) Domatic partition in homogeneous wireless sensor networks. J Netw Comput Appl 37:186–193
15. Censor-Hillel K, Ghaffari M, Kuhn F (2014) A new perspective on vertex connectivity. Proc SODA 2014:546–561
16. Zhang DQ, Li D (2014) High-density randomly deployed nodes sleep scheduling algorithm in wireless sensor networks. Adv Mater Res 846:446–451
17. Fuemmeler JA, Veeravalli VV (2008) Smart sleeping policies for energy-efficient tracking in sensor networks. In: Networked sensing information and control. Springer, USA, pp 267–287
18. Fuemmeler JA, Veeravalli VV (2010) Energy efficient multi-object tracking in sensor networks. IEEE Trans Signal Process 58(7):3742–3750
19. Fuemmeler JA, Atia GK, Veeravalli VV (2011) Sleep control for tracking in sensor networks. IEEE Trans Signal Process 59(9):4354–4366
20. Atia GK, Veeravalli VV, Fuemmeler JA (2011) Sensor scheduling for energy-efficient target tracking in sensor networks. IEEE Trans Signal Process 59(10):4923–4937
21. Jiang B, Ravindran B, Cho H (2013) Probability-based prediction and sleep scheduling for energy-efficient target tracking in sensor networks. IEEE Trans Mob Comput 12(4):735–747

22. Mihaylov M, Le Borgne YA, Tuyls K, Now A (2013) Reinforcement learning for self-organizing wake-up scheduling in wireless sensor networks. In: Agents and artificial intelligence. Springer, Berlin, pp 382–396
23. Wu F, Zilberstein S, Chen X (2011) Online planning for multi-agent systems with bounded communication. Artif Intell 175(2):487–511
24. Bhandarkar SM, Arabnia HR (1995) The REFINE multiprocessor: theoretical properties and algorithms. Parallel Comput 21(11):1783–1806
25. Arabnia HR, Smith JW (1993) A reconfigurable interconnection network for imaging operations and its implementation using a multi-stage switching box. In: Proceedings of the 7th annual international high performance computing conference. The 1993 High Performance Computing: New Horizons Supercomputing Symposium, Calgary, Alberta, Canada, June 349–357
26. Arabnia HR, Oliver MA (1989) A transputer network for fast operations on digitised images. Int J Eurogr Assoc (Comput Graphics Forum) 8(1):3–12
27. Arabnia HR (1990) A parallel algorithm for the arbitrary rotation of digitized images using process-and-data-decomposition approach. J Parallel Distrib Comput 10(2):188–193
28. Shi Z, Beard C, Mitchell K (2009) Analytical models for understanding misbehavior and MAC friendliness in CSMA networks. Perform Eval 66(9):469–487
29. Shi Z, Beard C, Mitchell K (2013) Analytical models for understanding space, backoff, and flow correlation in CSMA wireless networks. Wirel Netw 19(3):393–409
30. Shi Z, Beard C, Mitchell K (2011) Competition, cooperation, and optimization in multi-hop CSMA networks. In: Proceedings of the 8th ACM symposium on performance evaluation of wireless ad hoc, sensor, and ubiquitous networks. ACM, pp 117–120
31. Shi Z, Beard CC, Mitchell K (2007) Misbehavior and MAC friendliness in CSMA networks. In: Proceedings of WCNC, pp 355–360
32. Shi Z, Beard C, Mitchell K (2008) Tunable traffic control for multihop CSMA networks. In: Proceedings of military communications conference, 2008, MILCOM 2008. IEEE, pp 1–7
33. Shi Z, Gu R (2013) Efficient implementation of particle Swarm optimization algorithm. Int J Soft Comput Math Control 2(4):1–13
34. Ragi S, Chong EKP (2013) Decentralized control of unmanned aerial vehicles for multitarget tracking. Int Conf IEEE Unmanned Aircr Syst 2013:260–268
35. Demchik V (2011) Pseudo-random number generators for Monte Carlo simulations on ATI graphics processing units. Comput Phys Commun 182(3):692–705
36. Halton JH (1960) On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numerische Mathematik 2(1):84–90
37. Gmez-Prez D, Hofer R, Niederreiter H (2013) A general discrepancy bound for hybrid sequences involving Halton sequences. Unif Distrib Theory 8(1):31–45
38. Levy BC (2008) Principles of signal detection and parameter estimation. Springer, New York, USA
39. Hoeffding W (1963) Probability inequalities for sums of bounded random variables. J Am Stat Assoc 58(301):13–30