Computer Communications 35 (2012) 1029–1037

Contents lists available at SciVerse ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

ORACLE: Mobility control in wireless sensor and actor networks

Kaoru Ota [a,b,*], Mianxiong Dong [a,b], Zixue Cheng [a], Junbo Wang [a], Xu Li [b], Xuemin (Sherman) Shen [b]

[a] School of Computer Science and Engineering, The University of Aizu, Aizu-Wakamatsu 965-8580, Japan
[b] Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada

ARTICLE INFO

Article history:
Available online 27 August 2011

Keywords:
Wireless sensor and actor networks
Mobility control
Event prediction
Energy efficiency

ABSTRACT

Mobile actor is a promising tool for wireless sensor and actor networks (WSANs) provisioning energy and time efficient data collection. In this paper, we study actors' mobility control in WSANs for efficient events detecting in terms of time and energy consumption. We introduce an innovative approach ORACLE, to make actors predict events before sensors detection and migrate to the areas where the event may occur. In specific, we propose an event prediction scheme to predict an event from collected sensory data by utilizing the maximum likelihood estimation. Based on the perception, we design a control policy of actor's mobility pattern with Markov decision process. ORACLE not only enables minimal motion of actors which conserves time and energy to reach the event areas but also is energy-efficiency for sensors to reduce the forwarding range for event detection message. We evaluate the effectiveness of our proposed scheme through extensive experimental analysis.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

We have witnessed the rapid development of wireless sensor technologies recently because of the flexibility, self-organization, and low cost of conventional sensors. The more sensors in a network, the more efficient monitoring environments and processing sensory data is necessary for conserving both lifetime and energy of the network [1,21]. Introducing actors into the network, called wireless sensor and actor networks (WSANs), successfully accomplishes this requirement to collect and process data from sensors in the network [2–5,22]. In WSANs, sensors are low-cost and low-power and deployed throughout a field to sense environments, while actors are powerful and resourceful and are deployed much less than the number of the sensors. The actors collect and process data reported by the sensors and perform actions according to situations when the sensors detect events in the monitored field.

In general, the application scenarios in WSANs can be categorized basically into two types: complement of existing sensor networks and new application paradigms. The first category includes application scenarios where actors are employed for complementing/enhancing traditional sensor networks. For example, actors work as mobile sinks to achieve energy- and time-efficient data collection in the network [6]. Also, actors are used as energy suppliers to recharge energy in the battery of deployed sensors. Another category includes application scenarios where actors are employed in a new way such that actors react to events and work for "mission" accomplishment in event areas. For example, wildfire is detected by sensors deployed in a forest while actors receive sensor reports including a fire location and its intensity and actively move to extinguish a fire. Another example is, when a disaster happens (e.g., earthquake and flooding) and some people are missing, sensors find locations of victims and safe routes and accordingly actors guide a rescue team to the victims along the routes through a dangerous area [8]. In this paper, we focus on WSANs for the new application paradigms and assume an action area of actors coincides with an event area or is very close to the event area.

In literature, however, actors always walk randomly around the network before they receive event reports from sensors. It may cause not only data-delivery delay but also energy consumption on the sensors if the location of the actors is far from the location of the sensors nearby event areas. This is because many sensors are involved in relaying data to the actors which results in consuming enormous amounts of energy over the network. Also, if the actors take quite long time to migrate to action areas due to the long distance between the action areas and the location of the actors, such a waste of time could inflict considerable damage on mission-critical applications. Especially for applications dealing with emergent situations (e.g., lifesaving application), it is important for the actors to take less time to move to action areas after detecting events. An intuitive solution is to relax criteria of detecting events (e.g., decreasing a threshold) so that actors could arrive at action

* Corresponding author at: School of Computer Science and Engineering, The University of Aizu, Aizu-Wakamatsu 965-8580, Japan. Tel.: +81 519 729 2711.

E-mail addresses: k.ota@ieee.org, d8102104@u-aizu.ac.jp (K. Ota), d8101104@u-aizu.ac.jp (M. Dong), z-cheng@u-aizu.ac.jp (Z. Cheng), j-wang@u-aizu.ac.jp (J. Wang), xli@bbcr.uwaterloo.ca (X. Li), xshen@bbcr.uwaterloo.ca (X. (Sherman) Shen).

0140-3664/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2011.08.008

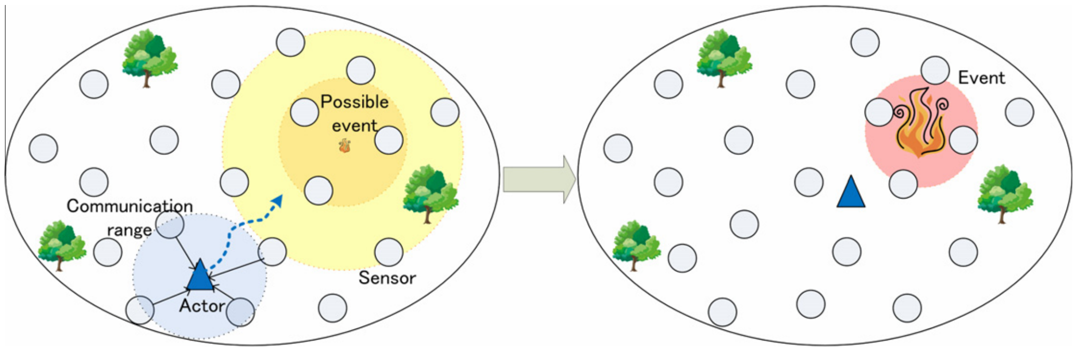

Fig. 1. Application example: sensors monitor wildfire and an actor predicts an event from sensor reports and moves toward the possible event [left] and finally is close to a location where the event occurs [right].

areas just on time when events occur. However, it might cause more energy consumption on sensors since the sensors are too sensitive and frequently detect events and report to actors a number of times. Further more, it might increase a false alarm rate of event detection which causes the waste of network bandwidth and energy. We argue that such problems can be solved by actor mobility control which makes actors to predict event areas and move toward the areas before the events actually occur. Hence, our objective is to design effective control of actor mobility to maintain quality of service, especially in case of emergent situations, while saving the energy on sensors and the time for the actors to migrate after the events.

To this end, first, we propose event prediction scheme for an actor to predict an event from collected sensory data by using the maximum likelihood estimation. The actor observes sensor readings along the way to move around the network before any event occurs. The actor can only encounter a subset of sensors during a short time with partial observations of the network, and estimate how likely an event occurs under the current condition. Second, we design a control policy of actor mobility with reinforcement learning in Markov decision processes. The actor determines next direction to migrate toward possible event areas by following the control policy based on the event prediction. Fig. 1 illustrates one of application examples, i.e., a WSAN is deployed for wildfire monitoring. Sensors monitor wildfire and report sensory data to an actor when they are in its communication range and the actor moves toward a possible fire by following the mobility control policy (left hand of Fig. 1). After some time periods, the actor gets close to a location on fire and extinguishes the fire immediately before it spreads (right hand of Fig. 1).

The remainder of our paper is organized as follows. Section 2 reviews related works. In Section 3, we propose the event prediction scheme. In Section 4, we design the actor mobility control policy based on the event prediction scheme. Section 5 evaluates the performance of our proposed control policy and presents the results with some discussion. Finally, we give concluding remarks and outline the directions for future work in Section 6.

2. Related work

Research issues on WSANs have been widely addressed by many articles recently [2–12]. Selvaradjou et al. [11] formulate the problem of optimal assignment of mobile actors in WSANs as Mixed Integer Non Linear Program, in order to conserve the energy needed for actor mobility but otherwise fulfill the deadline and resource constraints of events. Akyildiz and Kasimoglu [2] introduce lots of open research challenges for sensor-actor and actor-actor coordination and communication problems in WSANs. Melodia et al. [10] handle the coordination problem in WSANs. They present a coordination framework and developed an optimal solution for the sensor-actor coordination based on an event-driven partitioning paradigm and formulated it as an ILP (Integer Linear Problem). Melodia et al. [8] also propose a location management scheme to localize actors with minimal energy expenditure for sensors and design algorithms to deal with network congestion for traffic generated in the event area and with modeling actors' mobility for optimal task accomplishment based on the characteristic of the events. Li et al. [12] propose a novel localized algorithm, iMesh, which uses no global computation and generates constant per node storage load to deliver Distance-sensitive Service Discovery in WSANs. A route design problem for actors has been addressed to minimize their average inter-arrival time to sensors in [9]. The authors prove the problem is NP-hard and propose an optimized solution for the actors to effectively collect data from every sensor in the network by using minimum spanning trees.

However, those papers only consider the actors' motion after the events occur. To the best of our knowledge, this is the first work to deal with controlling actor mobility before the events occur. Our paper aims effective actor's mobility control which makes the actor migrate to event areas in a short time after sensors detect events by accurately predicting the event areas.

3. Event prediction scheme

In this section, we propose an event prediction scheme for an actor to predict an event under the current condition as well as to get a clue to the decision on whether the actor is on a "right" way to a possible event area. The actor observes sensory data from a subset of sensors and estimates a probability of how likely an event occurs from these partial observations using maximum likelihood estimation (MLE). The obtained probability is used for the actor mobility control in Section 4. First, we briefly describe the network model and then present how the event prediction scheme works in details.

3.1. Network model

A typical WSAN is composed of a large number of sensors Ns and some actors Na such that $Ns \gg Na$. The actors are deployed and move randomly around the network at the beginning of surveillance. An actor collects sensory data from every sensor encountered within a time window whose size is determined in advance. The time window is shifted by a unit of time. Data collected from n sensors in the time window is stored as the $1 \times n$ vector:

$$x(\Delta T) = [x_1, x_2, \dots, x_n] \tag{1}$$

where x_n is the sensory data of nth visited sensor within the time window. The data in the vector is deleted when the data is already out of the time window. This is because data in the vector should be temporal- and spatial-correlated each other to derive a probability of an event in terms of time and space (specified in the next section). On the other hand, the sensors are densely deployed on a sensing field and monitor environmental phenomenon and periodically store sensory data on themselves. Each sensor's communication range is very limited and the sensors can communicate only with their neighbors so that a sensor can transmit data directly to the actor only when the actor is within the sensor's range. When the sensor detects an event, it routes its information to the closest actor. According to the current trend, we apply a geographical routing paradigm to the sensor-actor communication [12,13]. Hence, the sensors require actors' location management and we apply an energy-aware solution [8] to the network model and its location management scheme takes advantages of two strategies: location updating and location prediction. Actors' location updates are broadcasted in limited ranges using Voronoi diagrams while actors' locations are predicted at the sensor-side based on previously received updates using Kalman filtering. An actor broadcasts location updates in its Voronoi cell if the actor's actual location is far from a location predicted by sensors within the cell.

3.2. Maximum likelihood estimation (MLE)

We systematically show how to utilize the MLE for an actor to predict an event in this section. The MLE is a widely used statistical method to provide estimates of a statistical model's parameters from given data such that the estimates fit the model. The MLE has been applied to several research fields which include signal processing in sensor networks [14,15].

We use spatial correlation among sensors to apply the MLE. For the dense sensor network, sensory data is highly spatial-correlated because sensors are close to each other [16]. From this point of view, such sensory data can be described by a normal distribution and the data distribution is different on a probability of an event. For example, in fire detection application, each sensor collects temperature of surroundings and the probability of developing fire is high in an area where the mean of temperature is higher and its variance is less than that of other areas. Consider an actor estimates the mean and the variance with the MLE by a set of sensory data collected on a route and derives the probability of an event. Assume the probability of the event can be determined by the mean and the variance of a unit area which is too large for the actor to observe in a short time period. In practical use, it is not difficult to obtain such a function for the probability of the event by computer simulations and/or real experiments. Then, the goal becomes to estimate the mean and variance in the vicinity of the actor's current location from its partial observations.

To this end, we first define parameter θ such as

$$\theta = (\mu, \sigma^2) \tag{2}$$

where parameter μ is the mean and σ^2 is the variance in the vicinity of the actor's current location. The actor wants to estimate θ from collected sensory data $x(\Delta T)$ in time window ΔT. $x(\Delta T)$ is collected from closely deployed sensors so as described by the normal distribution. The normal distribution $N(\mu, \sigma^2)$ has a probability density function which is

$$f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{3}$$

Sensory data in $x(\Delta T)$ is independent and identically-distributed and the corresponding probability density function can be given by

$$\begin{aligned} f(x(\Delta T)|\mu, \sigma^2) &= \prod_{i=1}^{n} f(x_i|\mu, \sigma^2) \\ &= \left(\frac{1}{2\pi\sigma^2}\right)^{n/2} \exp\left(-\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{2\sigma^2}\right) \end{aligned} \tag{4}$$

Thus, we maximize the following likelihood.

$$L(\theta) = f(x(\Delta T)|\theta) \tag{5}$$

To maximize the Eq. (5), we maximize its logarithm, called the log likelihood, which monotonically increases over the range of the likelihood. We differentiate the log likelihood with respect to μ and σ and equate to zero, respectively. The differentiated log likelihood with respect to μ can be described as

$$\frac{\partial}{\partial\mu} \log\left(\left(\frac{1}{2\pi\sigma^2}\right)^{n/2} \exp\left(-\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{2\sigma^2}\right)\right) = 0 \tag{6}$$

Eq. (6) can be rewritten for ease of computation as

$$\frac{\partial}{\partial\mu} \log\left(\left(\frac{1}{2\pi\sigma^2}\right)^{n/2} \exp\left(-\frac{\sum_{i=1}^{n}(x_i-\bar{x})^2 + n(\bar{x}-\mu)^2}{2\sigma^2}\right)\right) = 0 \tag{7}$$

where $\bar{x}$ is the mean of the collected sensory data. Then, we have

$$\hat{\mu} = \bar{x} = \sum_{i=1}^{n} x_i/n \tag{8}$$

Similarly, we maximize the log likelihood with respect to σ and have

$$\hat{\sigma}^2 = \sum_{i=1}^{n}(x_i-\hat{\mu})^2/n \tag{9}$$

Substitute $\hat{\mu}$ into Eq. (9) and finally we obtain the maximum likelihood estimator as

$$\hat{\theta} = (\hat{\mu}, \hat{\sigma}^2) = \left(\bar{x}, \sum_{i=1}^{n}(x_i-\bar{x})^2/n\right) \tag{10}$$

The actor calculates the probability of an event which depends on the mean and the variance of sensory data, respectively. Assume probability density functions of the mean and the variance are given in advance by computer simulations and/or real experiments. If the probability density function of the mean is denoted as $f_M(\mu)$ and the probability density function of the variance is denoted as $f_V(\sigma^2)$, the joint probability density function can be written as $f_{M,V}(\mu, \sigma^2)$. The actor can obtain the probability of the event when the mean μ and the variance σ^2 are satisfied simultaneously and then the current probability of the event $\Pr_t$ can be computed by

$$\Pr_t(a \leqslant M \leqslant b, c \leqslant V \leqslant d) = \int_a^b d\mu \int_c^d f_{M,V}(\mu, \sigma^2) d\sigma \tag{11}$$

where a, b, c, and d are constant values and show each range which values of (M, V) fall within. The actor finds the ranges for the mean $\widehat{\mu}$ and the variance $\hat{\sigma}^2$ obtained by Eq. (10) and finally calculates the probability of the event by Eq. (11).

4. ORACLE: Actor mobility control scheme

In this section, we propose the actor mobility control scheme, ORACLE, for the actor to decide a direction of migration while observing sensors before any event occurs. A main idea is as follows: (1) The actor compares the current probability of the event $\Pr_t$ to the previous probability $\Pr_{t-1}$ which is calculated with sensory data in time window $\Delta T - 1$; (2) The actor decides where to migrate by following a control policy based on a current state of whether the probability $\Pr_t$ is higher or lower than $\Pr_{t-1}$. The actor is motivated by a reward to be given if the probability $\Pr_t$ is higher than the previous one and the control policy shows the actor a direction where it is high possible for the actor to obtain the reward; and (3) The actor migrates to a new location, observes sensory data, and calculates the current probability. Then, it repeats the procedures (1) to (3) until it receives event reports from sensors. We achieve those procedures using reinforcement learning in Markov decision processes (MDPs) described in Section 4.2. We first define a direction where the actor migrates at the next time slot in the following section before going onto details of MDP based actor mobility control.

4.1. Definition of direction

The actor can observe sensory data only when it is within a transmission range of a sensor. It is important to effectively select sensors to be visited since unnecessary visiting results in consuming both energy on sensors and time for the actor to migrate. Thus, we define a direction as one of neighboring nodes where the actor migrates at the next time slot. The actor selects one node based on features of each neighbor as criteria to make a decision. For example, neighbor i produces more sensory data than the others and neighbor j maintains low error rate in its sensory data. For the actor to decide the direction without finding specific sensors, we define a feature of each sensor, called the *intensity* of a sensor, to show the sensor's surrounding information by involving its neighboring nodes. The intensity quantifies information produced by a sensor and its neighbors and shows how diverse data the information includes. For example, if sensors observe temperature of an environment, the intensity of a sensor is low when the sensor and its neighbors all produce the similar temperature values. Intuitively, when the actor finds the probability of the event increases at the current sensor, it may prefer to select a neighbor whose intensity is low since sensors around the neighbor possibly produce the similar data to the current sensor's data. However, only the intensity is not enough to decide which neighbor is the best to select, then other related conditions should be considered. Thus we introduce the MDP to the actor mobility control which is described in Section 4.2.

For calculating the intensity on each sensor, a sensor exchanges its own sensory data with its neighbors periodically but much less frequently than sensing environments. The sensor calculates its own intensity with all received data in a period of time and stores it locally and temporarily. The intensity is recalculated when new data comes in another period of time in order to keep temporal correlation among sensory data. The intensity is computed using entropy as follows.

Let X be the random variable representing a certain type of sensory data (e.g., temperature) from sensor i and its neighbors. For the sake of computation, we discretize the continuous sensory data values into Q disjoint intervals. For example, if the temperature value is 15 degrees Celsius and the temperature values are separated by five, we say the temperature is three. If we observe the sensors for N time slots in a period of time ΔT, the time series of the sensory data can be denoted by $D_{\Delta T} = (d_0, d_1, \ldots, d_{N-1})$ where $d_t \in [0, Q-1]$, $0 \leqslant t \leqslant N-1$ is the sensory data in time slot t. Assume each of these Q data values appears m_v times in $D_{\Delta T}$, $0 \leqslant v \leqslant Q-1$. Thus, the probability of the sensory data on the node being equal to particular value v can be computed as m_v/N. Therefore, the entropy of X is:

$$H(X) = -\sum_{v=0}^{Q-1} \frac{m_v}{N} \log \frac{m_v}{N} \tag{12}$$

The entropy is denoted as the intensity and the intensity of node i at time t can be described as

$$Intensity_i(t) = H(X) \tag{13}$$

Note that each sensor computes the intensity based on all sensory data it holds, which may consist of its own data and data received from a subset of all its neighbors. For example, a few or more data come from one neighbor and none of data comes from another neighbor. A sensor can compute the intensity without waiting for all neighbors to finish forwarding their sensory data to the sensor. Hence, a priority of exchanging packets for computing the intensity is lower than that of transmitting packets for notifying an event to the actor.

4.2. Reinforcement learning in Markov decision processes

We design actor mobility control using reinforcement learning in MDPs. The MDP is a well-known mathematical framework to dynamically formulate optimization problems of stochastic systems such as a queuing model, an inventory model, a reliability system, and so forth. More precisely, the MDP provides a decision-making model for changeable environments. When a process is in state s and a decision-maker takes action a which is available in state s, the process's state changes stochastically from state s to a new state s' at the next time step. The decision-maker obtains corresponding reword r on given action a and state s'. The decision-maker follows *policy* π which shows an action to be taken under the current state.

The MDP has been widely adopted for sensor network research [17,18]. One is target tracking in wireless sensor networks which accomplishes energy conservation of sensors by effectively predicting a target's trajectory using a Hierarchical MDP [17]. On the other hand, a ferry's mobility control framework in mobile ad hoc networks is designed based on an extended MDP called Partially Observable MDP (POMDP) [18], which aims for a data ferry to encounter as many as randomly moving sensors by controlling the ferry's motion using the POMDP. Our research is inspired by this research but we apply the MDP instead of the POMDP since the actor can directly observe the current state by the event prediction scheme presented in Section 3.

The MDP is represented by a 4-tuple $(S, A, P_\cdot(\cdot,\cdot), r(\cdot,\cdot))$ where,

- S is a set of states and state s_t denotes an increase or a decrease in the probability of the event at current time step t comparing to the previous one at previous time step $t-1$. Thus, S holds two states such that $S := \{increase, decrease\}$.
- A is a set of actions and action a_t denotes one of neighboring sensors where the actor will migrate at current time step t. Based on the intensity of the neighboring sensors, A includes three sensors whose intensity is the highest, medium, and the lowest in all neighbors respectively. The actor will choose one of those three sensors to migrate so that the set of actions can be $A := \{j_h, j_m, j_l\}$.
- $P_\cdot(\cdot,\cdot)$ is the state transition function represented by $P_a(s,s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$ which is the probability of state s' at next time step $t+1$ if the actor chooses action a in state s at time step t.
- $r(\cdot,\cdot)$ is the immediate reward represented by $r(s,a)$ which is the reward the actor obtains when the actor takes action a after that it observes state s. Here we set $r(s,a) = 1$ when $s = increase$ otherwise $r(s,a) = -1$.

The main objective is to find a policy π for the actor to migrate to next sensor (action) under the current probability of the event (state). π is a function specifying the action $\pi(s)$ to be selected when the current state is s. Whenever the actor selects an action and then a state is changed, the actor obtains a reward $r(s,a)$ based on given state s and action a. The goal is to choose a policy π that will maximize some cumulative functions of the rewards which can be denoted by $R(s_T, a_T)$ where s_T and a_T are a history of states and actions over a potentially long time window T respectively. That is formulated as:

$$\pi^* = \arg\max_{\pi} \mathrm{E}[R(s_T, a_T)|a_t = \pi_t(s_{t-1}, a_{t-1})] \tag{14}$$

Although the actor's mobility is modeled as Markovian such that it decides the next sensor to migrate only depending on a current state, the actor does not know the exact probabilities of the state transition $P_a(s,s')$. Then, the problem becomes a reinforcement learning problem in the MDP. The reinforcement learning is an area of machine learning to deal with a problem for an actor to decide an action with observations from an environment and to learn a policy capable to maximize rewards by series of actions [19]. A basic idea of the reinforcement learning is the actor repeats trials of taking several actions with several states and figures out an optimal policy from the experience.

For effective learning of the actor, a state-value function is defined to evaluate each state where the actor expects to be given rewards afterwards:

$$V^\pi(s) = \mathrm{E}_\pi(r_{t+1} + \gamma r_{t+1} + \ldots |s_t = s) = \mathrm{E}_\pi\left(\sum_{k=1}^{\infty} \gamma^{k-1} r_{t+k+1}\right) \tag{15}$$

where γ is the discount rate ($0 < \gamma \leqslant 1$) to evaluate a reward given in the future which is discounted over time. This is because a farther-future reward is less guaranteed since the environment surrounding the actor may change over time and the actor may not get a constant reward.

Eq. (15) considers only the impact of a current state on expected cumulative-rewards. However, in our scenario, the actor also considers the impact of an action selected when in the current state on the rewards. More specifically, when the probability of the event increases/decreases at a currently-visited sensor, it is a key to figure out which neighbors should be selected in order to make more rewards in the future. For this purpose, an action-value function is defined to evaluate a pair of a state and an action:

$$\begin{aligned} Q^\pi(s,a) &= \mathrm{E}_\pi(r_{t+1} + \gamma r_{t+1} + \cdots |s_t = s, a_t = a) \\ &= r(s,a) + \gamma \sum_{s'} V^\pi(s') P_a(s,s') \end{aligned} \tag{16}$$

Eq. (16) implies the action-value function will give an expected value of the cumulative rewards called a *Q-value* when the actor simply tries action a in state s and then follows policy π. An optimal action-value function which satisfies the above action-value function for every pair of a state and an action is described as:

$$Q^*(s,a) = \max_a \left[r(s,a) + \gamma \sum_{s'} Q^*(s',a) \right] \tag{17}$$

From Eq. (17), the actor will consider action a as an optimal action in state s when its Q-value is maximized. In other words, the policy $\pi(s)$ can be determined by the optimal action-value function Q^*, which is a precise estimate of the action-value function Q^π in Eq. (16). Then, the question is how to update a Q-value during the actor's experience in order to finally obtain the optimal-value function Q^*.

We apply a reinforcement learning technique called Q-learning to the problem [19]. The basic idea of Q-learning can be described as follows. When the actor takes an available action a_t in state s_t at time step t, Q-value $Q(s_t,a_t)$ is updated with $\max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$ where the actor is expected to choose action a_{t+1} which has the maximum Q-value in state s_{t+1}. Such a procedure can be formulated for any combination of states and actions as:

$$\begin{aligned} Q(s_t,a_t) &\leftarrow Q(s_t,a_t) + \alpha \\ &\quad \times \left[r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t,a_t) \right] \end{aligned} \tag{18}$$

where α is the learning rate ($0 \leqslant \alpha \leqslant 1$) which indicates how much the most recent information is included to the Q-value (e.g., $\alpha = 0$ means the Q-value will be never updated). Q-learning means that the Q-value converges to an optimal Q-value with probability one if every action is chosen by the actor a number of times although these actions are chosen in a random way. However, the actor is expected to acquire as many as rewards before obtaining the optimal Q-value in order to get close to a possible event area. In other word, the actor requires to gain experience efficiently to obtain the optimal Q-value in the shortest possible time. Therefore, some action-selection rules are proposed [19] and we apply the mostly used ε-greedy action-selection rule. In the ε-greedy rule, an action with a maximum Q-value at time step t is selected while the other actions are randomly selected with small probability ε. This can utilize estimated Q-values obtained from learning and at the same time efficiently seek better solutions for future.

Consequently, the actor determines a direction to migrate in the network while predicting an event from collected sensory data according to the following nine phases. Fig. 2 shows a flowchart of the actor mobility control scheme and the numbers in the figure are corresponding to the number of each phase.

(1) Initialize Q-values for every entry of $S \times A$
(2) Collect data from a sensor nearby the actor (current visited sensor) and calculate the probability of the event Pr_t at current time step t using MLE.
(3) The probability is compared to the probability Pr_{t-1} computed at previous time step $t-1$, and then observe a current state s_t and obtain a reward r_{t-1} according to s_t.
(4) Update Q-value by Eq. (18).
(5) Request sensors within a communication range of the actor to send their intensity and update a set of actions A.
(6) Shift a time step from t to $t+1$ and migrate to one of neighbor sensors chosen by the ε-greedy action-selection rule.
(7) Repeat phases (2) to (6) until the actor obtains an optimal Q-value.
(8) After the actor obtains the optimal Q-value, phase (6) is replaced to the following phase (6)′ such that (6)′ *shifts a time step t to t + 1 and migrates to one of neighbor sensors chosen by the optimal action-value function Q**

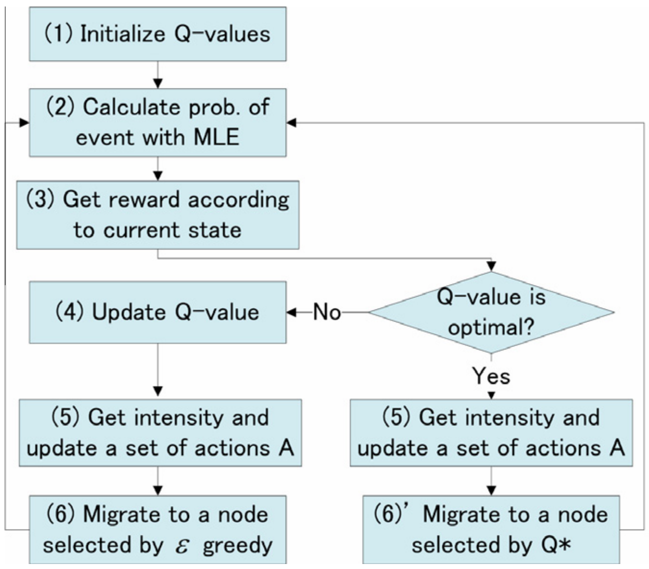
(1) Initialize Q−values
(2) Calculate prob. of event with MLE
(3) Get reward according to current state
Q−value is optimal?
No
Yes
(4) Update Q−value
(5) Get intensity and update a set of actions A
(5) Get intensity and update a set of actions A
(6) Migrate to a node selected by ε greedy
(6)' Migrate to a node selected by Q*

Fig. 2. The flowchart of the actor mobility control scheme ORACLE.

(9) Repeat phases (2), (3), (5), and (6)′ until the actor receives event reports from sensors.

5. Performance evaluation

In this section, we evaluate ORACLE by simulation experiments using Netlogo simulator [20]. We consider one actor dispatched to a monitored area and one event to occur in the area. Three key performance metrics in the experiments: distance, range, and energy consumption. The distance is between the actor and a sensor which first detects an event. The distance is shorter when the actor is closer to an event area, which implies the actor successfully predicts the event. This metric is used for all sets of experiments. The range indicates a moving range of the actor in the monitored area. It is ideal for the range to be minimized while the shorter distance is maintained. This metric is used in Section 5.1. The energy consumption indicates sensor energy consumed by introducing the event prediction scheme and additional contacts with the actor. It is important to reduce the energy consumption as much as possible since the sensor energy is battery-powered and very limited. This metric is used in the last section.

In our network settings, a number of sensors are randomly and densely deployed and the starting point of the actor is a fixed position. The event area is randomly chosen within the monitored area and environmental values are set on each sensor, which are spatially-correlated to the event area and change every unit of time. We randomly generate 50 network-examples for each network setting to obtain the distance, the range, and the energy consumption respectively from the average over those examples. The main parameters are set in the experiments as follows: the monitored area is from $80 \times 80\,\text{m}^2$ to $160 \times 160\,\text{m}^2$ in each set of experiments. A transmission range of sensors is 10 m and that of the actor is the same in this experiments. We use ρ to denote the rate of changing environmental values and two situations are considered: changeable and unchangeable environments. We set $\rho = 0.5$ and 0.02, respectively for the two situations. The actor migrates from sensor to sensor at one time step t and the environmental values change with ρ based on t. For example, when $\rho = 0.5$, the environmental values will change per two time steps. We use λ to denote the rate of exchanging data between sensors to compute the intensity for the event prediction scheme. We set $\lambda = 1, 0.3, 0.2$. For example, when $\rho = 0.5$ and $\lambda = 1$, sensory data is exchanged per two time steps. We set $\varepsilon = 0.01, 0.05, 0.1, 0.2$ for the probability used in the ε-greedy action-selection rule, respectively.

5.1. ORACLE vs. random walk model

In the first set of experiments, we compare ORACLE with a random walk model where the actor randomly migrates from sensor to sensor randomly. For random walk model, the actor “patrols” the monitored area without following any mobility control policy until it receives event reports from sensors. Other parameters such as the transmission range of the actor are the same for both models. In this experiments, we set the parameters $\rho = 0.5$, $\lambda = 0.3$, and $\varepsilon = 0.1$. Fig. 3 shows the distance from the event area to the actor over different monitored area by using the random model and ORACLE, respectively. The performance of the actor using ORACLE is better for a larger-scale network (160 square-meter area) since the actor has more opportunities to learn the optimal Q-value by encountering more various sensors. Fig. 4 shows the moving range of the actor. We can see that the actor can minimize migration by applying ORACLE while it effectively predicts the event area.

We also evaluate performance with different variances of environmental data observed by sensors. We assume data collected from closely deployed sensors is described by a normal distribution and the actor estimates parameter θ based on it as mentioned in Section 3.2. Thus, we investigate the impact of the variance on the performance in terms of the distance and the moving range. We vary the variance on the basis of a value used in the previous experiments and indicated as *sigma* on *x*-axis in Figs. 5 and 6 for simplicity. In this experiments, the monitored area is fixed to 120 square meter and other parameters are the same as the previous experiments. In Fig. 5, ORACLE still outperforms the random walk with changing the variance and however larger or smaller variance makes the performance worse since it may mislead the actor on predicting the probability of the event in some cases. On the other hand, the moving range of the actor using ORACLE is smaller with the smaller variance as shown in Fig. 6. The actor may require expanding a range to migrate in order to find the optimal Q-value when the variance is larger. However, ORACLE gets better performance than the random walk with every different variable.

From this set of experiments, we can conclude that the actor can successfully control its mobility by predicting event areas with minimal migration by using ORACLE even with different variables under the normal distribution. The scalability of ORACLE is also observed from the experiments.

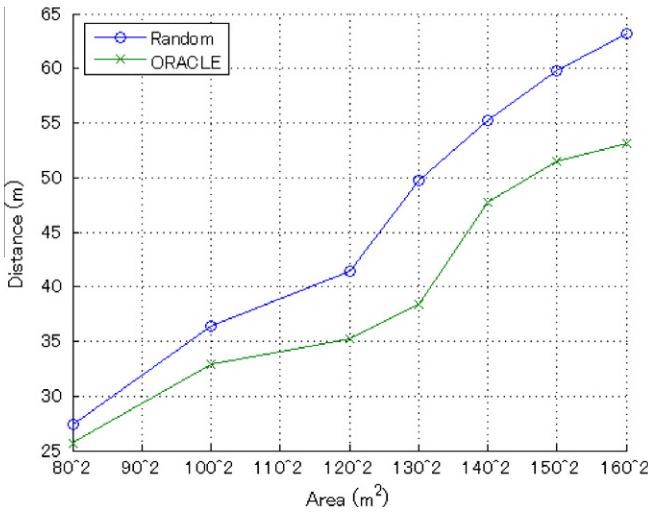

Fig. 3. Distance from the event area to the actor vs. monitored area.

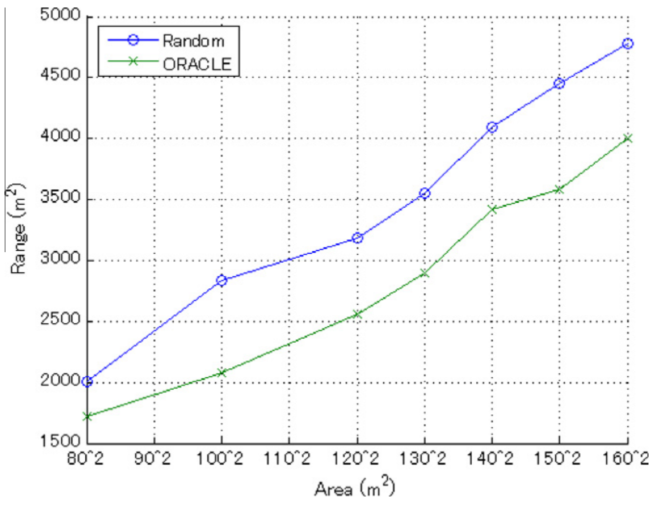

Fig. 4. Moving range of the actor vs. monitored area.

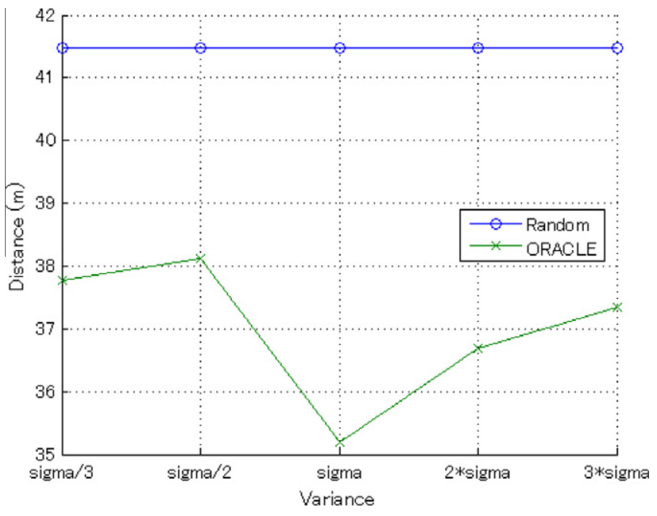

Fig. 5. Distance from the event area with different variances in 120 square-meter area.

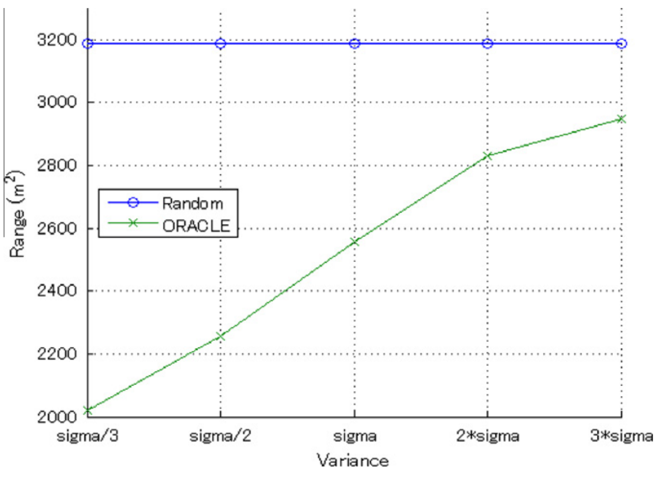

Fig. 6. Moving range of the actor with different variances in 120 square-meter area.

5.2. Impact of the rate of environmental change on the distance

In the second set of experiments, we evaluate ORACLE in terms of the distance under two different situations: changeable environment and unchangeable environment with $\rho = 0.5$, 0.02 respectively. The changeable environment means environmental values drastically change at each time step so that the event tends to occur in a short period. Likewise, the unchangeable environment indicates the opposite situation. The other parameters are fixed $\lambda = 0.3$ and $\varepsilon = 0.1$. Fig. 7 shows the distance from the event area to the actor with different rate of changing environments. The performance of the actor under the unchangeable environment is slightly better especially in a large-scale network. This is because the actor collects sensory data over a long period and estimates the optimal Q-value more precisely from accumulated experience. Experiment results indicate that ORACLE has the temporal-scalability as well as the spatial-scalability as shown in Section 5.1.

5.3. Impact of different values of the probability ε on the distance

In the third set of experiments, we evaluate ORACLE in terms of the distance with different probability used in the ε-greedy action-selection rule, $\varepsilon = 0.01$, 0.05, 0.1, 0.2, under the changeable and

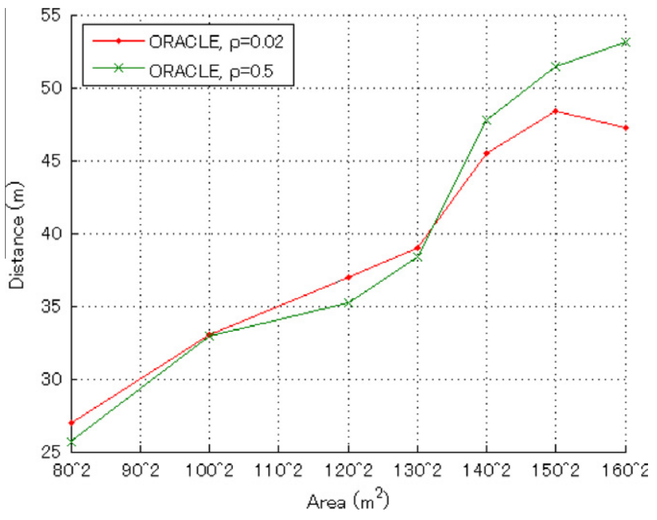

Fig. 7. Distance from the event area to the actor under two different environmental conditions.

unchangeable environments, respectively. The higher probability, the more every action has a chance to be selected in a random way similar to the random walk model. The data exchanging rate is fixed to be $\lambda = 0.3$, and the rate of environmental change is set as $\rho = 0.5$, 0.02 according to two different environmental conditions. Fig. 8 shows the distance from the event area to the actor with different probabilities under the changeable environment. Though there is no significant difference among performance with different probabilities, the performance of the actor is unstable when probability ε is high ($\varepsilon = 0.2$), such that the better performance is observed in the smaller-scale network while the performance gets worse in the large-scale network. On the other hand, under the unchangeable environment as shown in Fig. 9, the performance with the higher probability is better because information the actor can obtain from sensors may be limited in the unchangeable environment and it is helpful for the actor to seek better solutions by randomly selecting the next action.

The results of the experiments are useful to determine the parameter ε under different scenarios such as large- or small-scale network and a changeable or unchangeable environment. The probability between $\varepsilon = 0.05$ and $\varepsilon = 0.1$ is suitable for any scenario.

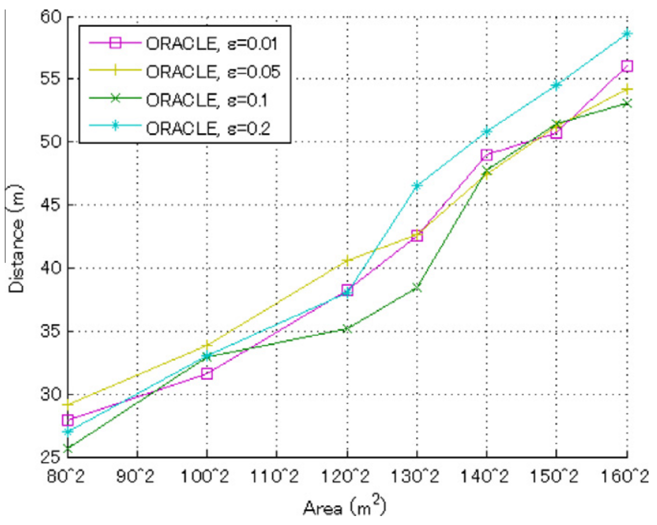

Fig. 8. Distance from the event area with different probability in changeable environment.

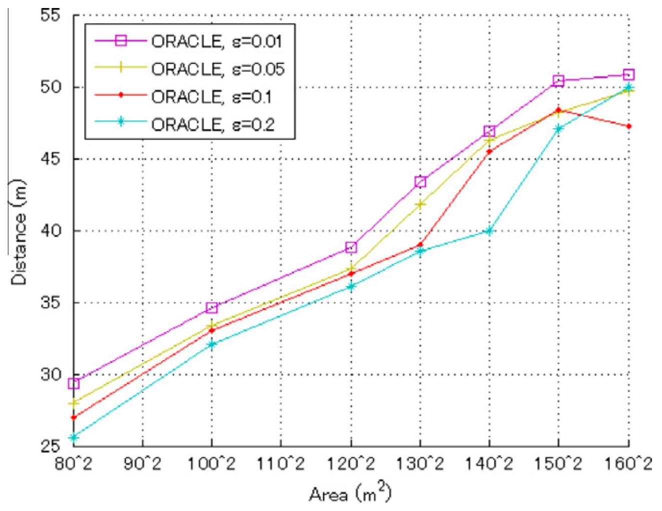

Fig. 9. Distance from the event area with different probability in unchangeable environment.

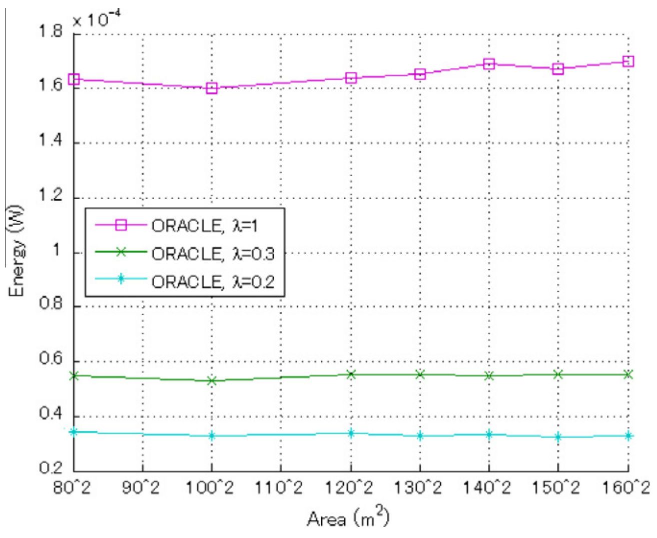

Fig. 11. Average energy consumption with different rate of exchanging data between sensors.

5.4. Impact of the rate of exchanging data between sensors on the distance and the energy

In the last set of experiments, we evaluate ORACLE in terms of the distance and the energy consumption of sensors by varying the rate of exchanging data between sensors, i.e., $\lambda = 1$, 0.3, 0.2, respectively. The energy consumption is calculated using the link metric [8]: $E = 2E_{elec} + E_{amp}d^{\alpha}$ where α is the exponent of the path loss propagation ($2 \leqslant \alpha \leqslant 5$), E_{amp} is a constant $[J/(bit \cdot m^{\alpha})]$, and E_{elec} is the energy for the transceiver circuitry to transmit or receive one bit $[J/bit] \cdot \alpha = 2$, $E_{amp} = 10p$, and $E_{elec} = 50n$ are used in this experiment. The other parameters are fixed as $\rho = 0.5$ and $\varepsilon = 0.1$.

As shown in Fig. 10, the exchanging rate has little impact on the performance of the actor. On the other hand, however, Fig. 11 shows the average energy consumption is much less when the exchanging rate is low. The results indicate that the sensor energy can be saved while the performance of the actor is maintained with the low rate of exchanging data among sensors. We can conclude that the actor can migrate to the event area both time-efficiently and energy-efficiently when the event occurs with ORACLE.

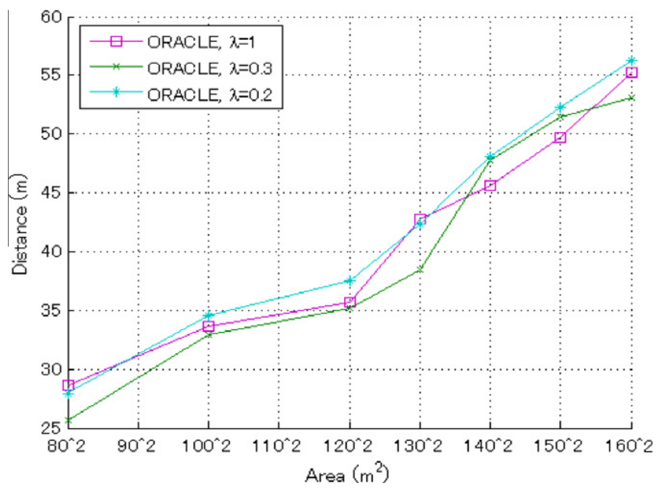

Fig. 10. Distance from the event area with different rate of exchanging data between sensors.

6. Conclusion

In this paper, we have proposed ORACLE which is a time and energy efficient scheme to control the actors' movement with event prediction. It has been demonstrated that with ORACLE (1) the actor can minimize migration comparing to the conventional actor mobility model while it effectively predicts the event area; (2) ORACLE has the temporal-scalability as well as the spatial-scalability; (3) appropriate parameter values are systematically found for better performance according to different situations; and (4) the sensor energy can be saved while the performance of the actor is maintained. For our future work, we will consider multi-actors' mobility control to deal with several events occurring simultaneously regarding velocity of each actor, intensity of each event, and mutual correlation of the both metrics.

Acknowledgements

This work is supported in part by Japan Society for the Promotion of Science (JSPS) Research Fellowships for Young Scientists Program, NEC C&C Foundation Grants for Non-Japanese Researchers, and the Circle for the Promotion of Science and Engineering Research Funding.

References

[1] K. Ota, M. Dong, J. Wang, S. Guo, Z. Cheng, M. Guo, Dynamic itinerary planning for mobile agents with a content-specific approach in wireless sensor networks, in: Proceedings of 2010 IEEE 72nd vehicular technology conference, Sep. 2010, pp. 1–5.

[2] I.F. Akyildiz, I.H. Kasimoglu, Wireless sensor and actor networks: research challenges, Ad Hoc Networks 2 (4) (2004) 351–367.

[3] X. Li, A. Nayak, D. Simplot-Ryl, I. Stojmenovic, Sensor Placement in Sensor and Actuator Networks, Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication, Wiley, 2010.

[4] J. Chen, X. Cao, P. Cheng, Y. Xiao, Y. Sun, Distributed collaborative control for industrial automation with wireless sensor and actuator networks, IEEE Transactions on Industrial Electronics 57 (12) (2010) 4219–4230.

[5] Xianghui Cao, Jiming Chen, Yang Xiao, Youxian Sun, Building environment control with wireless sensor and actuator networks: centralized vs. distributed, IEEE Transactions on Industrial Electronics 57 (11) (2010) 3596–3605.

[6] X. Li, A. Nayak, I. Stojmenovic, Sink Mobility in Wireless Sensor Networks, Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication, Wiley, 2010.

[7] X. Li, X. Liang, R. Lu, S. He, J. Chen, X. Shen, Toward reliable actor service in wireless sensor and actor networks, in: Proceedings of the 8th IEEE

international conference on mobile ad-hoc and sensor systems (MASS), Oct. 2011. (in press).

[8] T. Melodia, D. Pompili, I.F. Akyldiz, Handling mobility in wireless sensor and actor networks, IEEE Transactions on Mobile Computing 9 (2) (2010).

[9] Edith C.-H. Ngai, Jiangchuan Liu, Michael R. Lyu, Delay-minimized route design for wireless sensor-actuator networks, in: IEEE wireless communications and networking conference (WCNC'07), pp.3675–3680, Mar. 2007.

[10] T. Melodia, D. Pompili, V.C. Gungor, I.F. Akyildiz, Communication and coordination in wireless sensor and actor networks, IEEE Transactions on Mobile Computing 6 (10) (2007) 1116–1129.

[11] K. Selvaradjou, M. Dhanaraj, C. Siva Ram Murthy, Energy efficient assignment of events in wireless sensor and mobile actor networks, in: 14th IEEE international conference on networks 2006 (ICON '06), vol.2, pp.1–6, Sep. 2006.

[12] X. Li, N. Santoro, I. Stojmenovic, Localized distance-sensitive service discovery in wireless sensor and actor networks, IEEE Transaction on Computers 58 (9) (2009) 1275–1288.

[13] H. Zhang, H. Shen, Energy-efficient beaconless geographic routing in wireless sensor networks, IEEE Transactions on Parallel and Distributed Systems 21 (6) (2010) 881–896.

[14] I. Ziskind, M. Wax, Maximum likelihood localization of multiple sources by alternating projection, IEEE Transactions on Acoustics, Speech and Signal Processing 36 (10) (1988) 1553–1560.

[15] S. Barbarossa, G. Scutari, Decentralized maximum-likelihood estimation for sensor networks composed of nonlinearly coupled dynamical systems, IEEE Transactions on Signal Processing 55 (7) (2007) 3456–3470.

[16] S. Pattem, B. Krishnamachari, R. Govindan, The impact of spatial correlation on routing with compression in wireless sensor networks, ACM Transactions on Sensor Networks 4 (4) (2008).

[17] W.-L. Yeow, C.-K. Tham, W.-C. Wong, Energy efficient multiple target tracking in wireless sensor networks, IEEE Transactions on Vehicular Technology 56 (2) (2007) 918–928.

[18] T. He, K.-W. Lee, A. Swami, Flying in the dark: controlling autonomous data ferries with partial observations, in: Proceedings of the eleventh ACM international symposium on Mobile ad hoc networking and computing (MobiHoc '10), 2010, pp.141–150.

[19] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 1998.

[20] S. Tisue, U. Wilensky, Netlogo: A simple environment for modeling complexity, in: international conference on complex systems, pp.16–21, 2004.

[21] J. Chen, W. Xu, S. He, Y. Sun, P. Thulasiramanz, X. Shen, Utility-based asynchronous flow control algorithm for wireless sensor networks, IEEE Journal on Selected Areas in Communications 28 (7) (2010) 1116–1126.

[22] J. Chen, Q. Yu, P. Cheng, Y. Sun, Y. Fan, X. Shen, Game theoretical approach for channel allocation in wireless sensor and actuator networks, IEEE Transactions on Automatic Control, (in press).