Energy efficient transmission scheduling for infrastructure sensor nodes in location systems

Jongwook Lee\textsuperscript{a}, Saewoong Bahk\textsuperscript{a}, Jin-Ghoo Choi\textsuperscript{b,}\textsuperscript{*}

\textsuperscript{a}Department of EECS and INMC, Seoul National University, 599, Gwanangno, Gwanak gu, Seoul, Republic of Korea
\textsuperscript{b}Department of Information and Communication Engineering, Yeungnam University, 214-1 Dae-dong Gyeongsan-si Gyeongsangbuk-do, Republic of Korea

\textbf{Abstract}

This paper considers a location system where a number of deployed sensor nodes collaborate with objects that need to be localized. Unlike existing works, we focus on reducing the energy consumption of the sensor nodes, which are assumed to be static and run on limited battery power. To minimize the total wake-up time of the sensor nodes, we control the transmission schedule of each object. Because it is difficult to find an optimal solution to the considered optimization problem, we consider an approach to this problem that consists of two steps: (1) create an equivalent modified graph coloring subproblem, and (2) permute the coloring result to obtain a best possible solution. We adopt some existing graph coloring algorithms for step 1 and find two properties of optimal schedules that can be used to confine the search space for step 2. Additionally, we propose a heuristic algorithm that aims at significantly reducing the complexity for the case where the confined search space is still too large. The performance of our heuristic algorithm is evaluated through extensive simulations. It is shown that its performance is comparable to that of the simulated annealing algorithm, which gives a near-optimal solution.

© 2010 Elsevier B.V. All rights reserved.

\textbf{1. Introduction}

It is widely accepted that an important feature in the upcoming ubiquitous computing era will be location-awareness. This fact is witnessed by the recent growth of the multi-billion dollar global positioning system (GPS) industry. By receiving multiple signals from orbiting GPS satellites, a GPS receiver can triangulate itself to determine its precise position. However, GPS does not work well indoors because the GPS signal cannot travel far through a building wall. In addition, a number of applications have appeared that demand higher positioning accuracy than GPS can provide. For these reasons, a great deal of research has been focused on the development of location systems over the last few decades [1–4].

Location systems have some different characteristics from target tracking systems (see [5] for the classification of wireless sensor network (WSN) applications). In a target tracking system, a large number of sensor nodes are deployed randomly in an area of interest, such as a battlefield. With deployed sensor nodes that run on limited battery power, the target tracking system pursues two conflicting goals: (i) prolonging the system lifetime by allowing the sensor nodes to sleep as much as possible and (ii) satisfying the predetermined system requirements, such as tracking accuracy. Therefore, numerous studies have been conducted to find an optimal trade-off between the two goals or a way to meet these at the same time [6–9].

Unlike the target tracking system where each sensor node is required to adopt an adequate target sensing technique, sensor nodes and objects\textsuperscript{1} in a location system cooperate with each other to pinpoint the positions of the objects. In other words, they transmit/receive signals

\textsuperscript{1} The terminology differs depending on the localization technique and system used. In this paper, we use \textit{sensor node} to denote a static infrastructure device whose position is known, and \textit{object} to denote a mobile target that is to be localized.
to/from their neighboring counterparts periodically or on an on-demand basis. The actual calculation of the position can be performed either at a sensor node or at an object according to the localization technique employed.

Since the location system was introduced, most research has been focused on developing more accurate localization techniques \cite{10,11} (see \cite{12} for the taxonomy of localization techniques) or to minimize each object’s energy consumption \cite{13,14}. The energy consumption of sensor nodes is often ignored because sensor nodes are assumed to be mains-powered. Alternatively, it is assumed that the energy-saving mechanisms used in target tracking systems can be applied without modification. However, supplying power to sensor nodes can be expensive in certain circumstances. For example, a dense deployment of sensor nodes to enhance the localization accuracy will incur considerable installation costs. In this case, it is cost-effective to install battery-powered sensor nodes and periodically replenish or change their batteries. As a consequence, an energy-saving mechanism of sensor nodes is indispensable to minimize human intervention. In addition, the energy-saving mechanism in a location system distinguishes itself from that which is applied in a target tracking system because it can exploit cooperative communications between sensor nodes and objects.

In this paper, an optimization problem that aims at minimizing the total energy consumption of sensor nodes is presented. Throughout the paper, we consider a location system where each object transmits a signal periodically at some assigned time slot, and the sensor nodes in the vicinity of the object receive the signal to cooperatively localize the object’s position. That is, this scheme works by assuming that when a transmitter transmits a signal, multiple receivers exist. The transmitter (i.e., object), which should be extremely cheap and energy efficient for mass deployment, can benefit from this scheme. The goal of this paper is not to develop an accurate localization technique but to minimize the energy consumption of sensor nodes. The energy saving is accomplished by scheduling each object’s transmission, thereby minimizing the wake-up time of each sensor node. However, as it is difficult to find an optimal solution when the problem size is large, we first present two properties of optimal schedules that can be useful in reducing the search space. Then, we propose a heuristic algorithm to handle the case where the reduced search space is still too large. Through extensive simulations, we evaluate the efficiency of our heuristic algorithm and discuss some factors that affect its performance.

The rest of the paper is organized as follows: In Section 2, our considered system model is described. In Section 3, we precisely formulate the optimization problem. To solve the problem by decomposition, we propose to resolve a graph coloring subproblem with a slightly different objective from conventional graph coloring problems. We elaborate on the properties of optimal schedules and the details of the proposed heuristic algorithm in Sections 4 and 5, respectively. In Section 6, we explain the simulated annealing algorithm, which is employed to obtain a near-optimal solution to the optimization problem. Then, we compare it with our heuristic algorithm under various scenarios. Section 7 covers related work and is followed by concluding remarks in Section 8.

2. System model

2.1. General description

The considered location system aims to localize a large number of objects, denoted by $O(i \leq j \leq N_O)$, based on the information gathered by static sensor nodes, denoted by $S(i \leq i \leq N_S)$. Each object in the system is in either of two states: the stationary or the moving state. The state of an object determines when and how the object is localized. We assume that the objects in the system remain stationary most of the time. Such a scenario can be applied to warehouse inventories, libraries, and so on. Also, it is assumed that both the sensor nodes and the objects are battery-powered. In such a system, a stringent delay constraint that typical real-time location systems generally impose is of less significance. Rather, our concern is to minimize the sensor nodes’ energy consumption in localizing the stationary objects.

Unlike moving objects, each stationary object is assigned a time slot periodically for transmitting a short packet, called a blink, which is used for localizing the object. Among sensor nodes in the vicinity of a stationary object, one is designated as the object’s serving sensor node. The role of a serving sensor node is to control the blink transmission from each of its associated objects. In response to a received blink, it transmits an acknowledgment packet that carries all necessary information for assigning a new time slot to the object (coarse-grained control) or compensating for the object’s clock drift (fine-grained control).

We consider the time-difference-of-arrival (TDOA) multilateration as a localization technique for stationary objects. The TDOA technique estimates a signal emitter’s location by measuring the time-difference-of-arrival at three or more receiver sites. The time difference between any two receivers defines a hyperbola on which the signal emitter may exist, assuming that the signal emitter and the receivers are coplanar. If this procedure is done again with another receiver in combination with any of the previously used receivers, another hyperbola is defined and the intersection of the two hyperbolas results in the location estimate of the signal emitter \cite{15}. Therefore, tight time synchronization among receivers is essential for an accurate estimation in this technique. In our system, each stationary object acts as a signal emitter; it transmits a blink, which is received by several nearby sensor nodes. Before these sensor nodes receive the blink, they have been

2 Unlike most of the previous studies that aimed at tracking a small number of moving targets, we consider scenarios where there is a great number of objects to localize, e.g., hundreds of thousands of books in a library. In such scenarios, it is reasonable to assume that objects are stationary most of the time. Because our algorithm is designed for such scenarios, it becomes less efficient as the number of moving objects increases.

3 For the rest of this paper, we assume that a sufficient number of receivers for localization are deployed around each object in the system.
synchronized via a mechanism that will be described in the following subsection.

Each localization round consists of three periods: the beacon period (BP), localization period (LP), and contention period (CP), as depicted in Fig. 1. The behavior of sensor nodes and objects differs among the periods. In what follows, we explain all the periods in detail, focusing on the behavior of sensor nodes, stationary objects, and moving objects separately.

2.2. Beacon periods

A BP is composed of several time slots, called beacon slots, among which each sensor node is assigned one for its beacon transmission as in [16]. The purpose of using beacons is twofold: one is to achieve tight time synchronization among sensor nodes, and the other is to enable each object to detect its movement. According to the component type, each node or object operates as follows:

2.2.1 Sensor nodes

For tight time synchronization, all the sensor nodes in the system form a spanning tree rooted at a predetermined synchronization starting point. Because each sensor node is assumed to know the distance between itself and its parent node in advance, it can adjust its local clock based on the time-of-arrival (TOA) of its parent node’s beacon. To be specific, each sensor node compensates for its local clock drift whenever it receives a beacon from its parent node. Because the parent node is synchronized with the synchronization starting point when it transmits a beacon (this will be explained shortly), which is transmitted at the exact beginning of a beacon slot, the time difference between the expected and actual beacon reception times at the child node amounts to the clock drift of the child node. By adding the difference to its local clock, the child node can synchronize itself with its parent node.

Fig. 2 illustrates an example of the synchronization process. By adding $t_3 - t_4$ to its local clock, the child node can synchronize itself with its parent node.

2.2.2. Stationary objects

In every BP, each stationary object should wake up and listen to the beacon slot of its serving sensor node to check whether it is still in the communication range of its serving sensor node. If it receives the beacon, it can go to sleep until its assigned time slot in the following LP. Otherwise, its state is changed to the moving state.

2.2.3. Moving objects

Each moving object should wake up at the beginning of every BP and receive all beacons from neighboring sensor nodes to acquire their addresses. This is to facilitate the localization process, which is completed in the following CP.

2.3. Localization periods

LPs are used for localizing stationary objects. As mentioned earlier, each stationary object is assigned a time slot for blink transmission in every LP. In the rest of this paper, a time slot denotes the one in an LP, not a beacon slot, unless specified otherwise.

2.3.1. Sensor nodes

Each sensor node should wake up at the beginning of every LP and stay awake until all of its neighboring stationary objects transmit their blinks. Certainly, it is possible for

---

Footnote 4: This can be made possible by a two-way time-of-arrival measurement, for example.

Footnote 5: One possible approach for obtaining a feasible beacon schedule is to visit each sensor node in the tree according to breadth-first search (BFS) algorithm and assign an available beacon slot with the minimum index.
a sensor node to wake up, not at the beginning of an LP, but at the first time slot at which it is supposed to receive a blink. However, the additional energy saved by doing so is not significant. This is because the length of a schedule (defined in Section 4) cannot be arbitrarily long because the estimation accuracy deteriorates as time goes on due to the relative clock drift among sensor nodes. To shorten the length of a schedule, it is desirable to fully utilize the spatial reuse property of the wireless medium, which implies that the time difference between the two moments would not be so long.

Sensor nodes that received a blink from an object record the blink reception time. Because they are tightly synchronized in the preceding BP, the object’s position can be estimated later, in the following CP, based on their recorded times using the TDOA technique.

2.3.2. Stationary objects

At the assigned time slot in an LP, each stationary object wakes up and transmits a blink. After receiving an acknowledgement packet from its serving sensor node, it goes back to sleep until the next localization round begins.

To guarantee sufficiently high signal-to-interference-plus-noise ratio (SINR), which is vital for accurate estimation in TOA-based localization techniques, blinks must not interfere with each other, and collisions must not occur. Hence, a blink schedule should be made in such a way that any two objects in an interference relationship are not assigned to the same time slot in an LP. For brevity, we simply use the term “schedule” instead of “blink schedule” in the rest of this paper.

2.3.3. Moving objects

In this period, moving objects do nothing.

2.4. Contention periods

CPs are used for general purposes, such as the localization of moving objects, collaboration among sensor nodes, and communication to/from a central server. Sensor nodes and moving objects contend for channel access during CPs unlike in BPs and LPs that are based on time division multiple access (TDMA).

2.4.1. Sensor nodes

In this period, sensor nodes adjacent to a stationary object collaborate with each other to localize the object. Once the location is estimated, it is reported to the central server. However, it is possible, depending on the application, to execute this procedure either periodically or on an on-demand basis to reduce the communication traffic. To reduce energy consumption, sensor nodes operate in an energy-saving mode, e.g., alternating between sleep and wake-up states. Some well-known energy-saving MAC and routing protocols in the literature [17–20, 38, 39] can be adopted for this purpose.

2.4.2. Stationary objects

In this period, stationary objects do nothing.

2.4.3. Moving objects

Each object in the moving state has to localize itself using a localization technique, such as a two-way TOA measurement. Upon estimating its location, it reports this information to the central server via intermediate sensor nodes. If it is noticed that a moving object has stayed at a certain location longer than a predetermined threshold, the central server updates and redistributes the schedule including the object. Consequently, one of nearby sensor nodes is designated as the serving sensor node of the object, and the object’s state changes to the stationary state.

Fig. 3 shows an operational example of the considered system. The operations of two sensor nodes (s1 and s2) and two stationary objects (o1 and o2) are shown in the figure. Solid, shaded, and hollow rectangles represent transmission, reception, and idle listening, respectively. In the synchronization tree, s1 is assumed to be the parent node of s2. In addition, s1 and s2 are the serving sensor nodes of o1 and o2, respectively. Although not shown in the figure, the two sensor nodes have a number of neighboring stationary objects whose blinks are transmitted at their assigned time slots. In the CP, the sensor nodes have their own active duration for communication, which is similar to the approach used in [21].

2.5. Mathematical system model

Let S and S′ denote the sets of sensor nodes that are in the transmission and interference ranges of oj, respectively. Conversely, when si is located in the transmission ranges and interference ranges of some objects, the sets of objects are denoted by Oj and Oj′, respectively. By definition, Sj ⊆ Sj′ and Oj ⊆ Oj′. Also, O = ∪j∈SjOj = {o1, o2, …, on}, if there is no isolated object. In the rest of this paper, we use the terms neighboring objects of sj and neighboring sensor nodes of oj to denote Oj and Sj, respectively.

Each LP consists of K consecutive time slots, and K = {1,2,3,…,K} is the set of time slot indices. A schedule is then defined as follows:

---

6 In practice, it is hard to know the exact transmission and interference ranges used for blink scheduling. Therefore, these values should be configured somewhat conservatively considering the used wireless channel model.

7 It is possible that an object has changed its location but is still able to receive a beacon from its serving sensor node. Therefore, the sets should be obtained considering this fact.
Definition 1. A schedule \( \pi : K \rightarrow 2^O \) is a function that assigns a time slot to each object. It has the following properties:

1. \( \bigcup_k A_k = O \).
2. \( A_k \cap A_{k'} = \emptyset \) if \( k \neq k' \).
3. \( \forall o_m, o_n \in A_k, S_m \cap S_n = \emptyset \) if \( m \neq n \).
4. \( A_k = \emptyset \) for \( k > k_0 \) if \( A_{k_0} = \emptyset \).

where \( A_k = \pi(k) \) is the set of objects that are assigned to time slot \( k \).

Properties (1) and (2) together state that all the objects should be scheduled\(^8\) only once in an LP. Property (3) imposes a constraint that two objects cannot be assigned to the same time slot when they interfere with each other. This simple interference model is called the protocol model [22].\(^9\) Property (4) implies that when one of the empty time slots is to be assigned to a certain object, the time slot with the lowest index is assigned first.

According to the described behavior of a sensor node, when schedule \( \pi \) is applied, the wake-up time of \( s_i \) can be represented as

\[ w_i(\pi) = \max \{ k | o_j \in \pi(k) \text{ for some } o_j \in O_i \} \]  

Then, the total wake-up time of all the sensor nodes is

\[ w(\pi) = \sum_{i=1}^{N} w_i(\pi). \]  

Here, schedule \( \pi \) is a function of the LP because it is possible to have a different schedule for each LP. However, we omit the index of the LP because we focus on the scheduling aspects of a particular LP.

3. Problem formulation

In this section, we formulate an optimization problem for the described system model and present its equivalent modified graph coloring problem to handle the complexity of the optimization problem.

3.1. Optimization problem

To minimize the total energy consumption of the sensor nodes, we need to find an optimal schedule \( \pi^* \) that satisfies:

\[ \pi^* \in \arg \min_{\pi \in \Pi} w(\pi), \]  

where \( \Pi \) denotes the set of all feasible scheduling solutions. Note that the optimal schedule may not be unique because there can be more than one schedule that achieves the same minimum objective value.

We define an indicator variable \( a_{ijk} \) that equals 1 if \( o_j \) is scheduled to time slot \( k \), and 0 otherwise. In addition, we let \( b_i \) denote the wake-up time of \( s_i \). Then, the considered model can be expressed as an optimization problem whose objective is to minimize the sum of all the sensor nodes’ wake-up times:

\[ \min_{a_{ijk}, b_i} \sum_{i=1}^{N} b_i \]  

subject to

1. \( \sum_{k=1}^{K} a_{ijk} = 1, \quad 1 \leq j \leq N_o, \) \hspace{1cm} \( 1 \leq i \leq N_s, 1 \leq k \leq k_f, \) \hspace{1cm} \( 0 \leq a_{ijk} \leq 1, \) \hspace{1cm} \( \forall j \in O_i, 1 \leq i \leq N_s, 1 \leq k \leq k_f, \)

\[ \big| O_i - O_j \big| + \min_{\forall j_1 \in O_i, \forall j_2 \in O_j} a_{ij_1k} + a_{ij_2k} \leq \big| O_i - O_j \big|, \]  

\[ \sum_{k=1}^{K} k \cdot a_{ijk} \leq b_i, \quad \forall j \in O_i, 1 \leq i \leq N_s, 1 \leq k \leq k_f, \]  

\[ a_{ijk} \in \{0, 1\}, \quad 1 \leq j \leq N_o, 1 \leq i \leq N_s, 1 \leq k \leq k_f. \]  

Eq. (5) ensures that an object is scheduled only once in every LP. Eqs. (6) and (7) are derived from the interference constraint. According to (6), it is prohibited to have more than one object in \( O_i \) assigned to the same time slot. Additionally, in (7), if one of the objects in \( O_i \) is assigned to a certain time slot (i.e., the first term of the left-hand side equals to \( |O_i - O_j| \)), all objects that are in the interference range but not in the transmission range of \( s_i \) should not be assigned to the same time slot (i.e., the second term of the left-hand side should equal 0 to satisfy the inequality). Eq. (8) comes as a result of applying a common trick to (1). This converts a \( \min \max \) objective function into a simple \( \min \) objective function by introducing the decision variables \( b_i \). Lastly, (9) imposes integer constraints on the \( a_{ijk} \) variables. In general, it is known that the integer programming (IP) to which the formulated problem belongs is very hard to solve unless the constraint matrix is totally unimodular and the right-hand sides of the constraints are integers, which do not hold in our case.

3.2. Modified graph coloring problem

To manage the complexity of the formulated optimization problem, we convert it into a graph coloring problem with a slightly different objective from those used in conventional graph coloring problems. Consider an interference graph \( G = (V, E) \) with a vertex set \( V \) and an edge set \( E \). From the given topology,\(^10\) we can construct a graph as follows: (1) let \( o_j \) be the \( j \)-th vertex in \( V \), and (2) put an edge between the \( j_1 \)-th vertex and the \( j_2 \)-th vertex if \( S_{j_1} \cap S_{j_2} \neq \emptyset \) or \( S_{j_1}' \cap S_{j_2}' \neq \emptyset \).

Let \( c : V \rightarrow \{1, 2, \ldots, k_f\} \) be a vertex coloring, and \( V_k \) be the set of vertices that are assigned color \( k \) by coloring \( c \). If we define \( \pi(k) = \{o_{ij} \in V_k\} \), then it satisfies all the Properties (1)–(4) of Definition 1. Conversely, when the schedule \( \pi \) is given, the graph can be colored by choosing \( V_k = \{j | o_j \in \)
\( \pi(k) \). Therefore, there is a one-to-one relation between the two.

However, finding an optimal schedule is not exactly the same as coloring the graph with the minimum number of colors, namely the chromatic number \( \chi(G) \). Whereas traditional coloring problems are concerned with finding a way to color the given graph with a small number of colors, our problem needs further consideration. This is because, in our problem, each object affects at least one neighboring sensor node. In other words, the wake-up time of a sensor node depends on which time slots to which the neighboring objects are assigned. Consequently, to find an optimal solution, the neighbor relations between sensor nodes and objects need to be examined carefully.

3.3. Illustrative example

Fig. 4 shows an example of the modified graph coloring problem. In (a), which shows the topology, sensor nodes and objects are represented as circles and rectangles, respectively. Also, the neighbor relations are depicted as dotted lines. In (b) and (c), the number above or below each vertex (object) represents a color index, i.e., a time slot in our context.

**Fig. 4.** An example of the modified graph coloring problem. In (a), which shows the topology, sensor nodes and objects are represented as circles and rectangles, respectively. Also, the neighbor relations are depicted as dotted lines. In (b) and (c), the number above or below each vertex (object) represents a color index, i.e., a time slot in our context.

It is easy to find that the chromatic number of the given graph, i.e., \( \chi(G) \), is 4. In conventional graph coloring problems, the goal is to find a way to color the graph with \( \chi(G) \) colors. In that sense, both schedules 1 and 2 in Fig. 4(b) and (c) achieve the same goal. In the figures, the number above or below each vertex represents a color index, i.e., a time slot in our context. However, depending on which time slot each object is assigned to, the total wake-up time of the sensor nodes is different. It is 11(=4 + 4 + 3) when schedule 1 is applied, whereas in schedule 2 it is reduced to 9(=3 + 4 + 2). This simple example tells us that finding a scheduling algorithm that gives the minimum total wake-up time is not straightforward.

### 4. Properties of optimal schedules

Our approach to this problem consists of two steps: (1) solve the coloring subproblem, and (2) permute the coloring result to obtain the best possible solution.

As for the coloring subproblem, we employ a well-known coloring algorithm as in [24,25], which runs with reduced complexity\(^{11} \) at the cost of optimality.\(^{12} \)

\(^{11}\) The problem of finding a chromatic number is NP-hard. The corresponding decision problem – is there a coloring that uses at most \( k \) colors? – is NP-complete [23].

\(^{12}\) In Section 6, we demonstrate that, even though coloring a graph with a small number of colors is generally beneficial, an optimal coloring with \( \chi(G) \) colors does not necessarily result in an optimal schedule.
Once the coloring result is obtained, the permutation of the coloring result is performed. Let \( l(\pi) \) denote the length of schedule \( \pi \), i.e., \( l(\pi) = \max(k|\pi(k) \neq \phi) \), where \( \lambda(G) \leq l(\pi) < N_\phi \). For brevity, we use \( l \) instead of \( l(\pi) \).

Then, a brute force search for the permutation requires us to check the \( l! \) cases, which is beyond the realm of practicality. To confine the search space, we exploit the properties of optimal schedules.

Let \( \Pi_l \) and \( \pi_l^i \) denote the set of length-\( l \) schedules and the schedule that achieves the minimum total wake-up time among the schedules in \( \Pi_l \), respectively. In other words, \( \pi_l^i = \arg \min \{ l(\pi) | \pi \in \Pi_l \} \). Additionally, let \( \Pi_l^i \) denote the set of \( \pi_l^i \)'s. We arrange the wake-up times of the sensor nodes in ascending order when schedule \( \pi \) is applied, resulting in \( w(1) < w(2) < \cdots < w(q_M) \), where \( q_M \leq N_\phi \). Furthermore, we define \( S(q) = \{ s_i | w(\pi) = w(q), 1 \leq q \leq q_M \} \) \( S(q) = \{ k | w(q-1) + 1 \leq k < w(q), 1 \leq q \leq q_M \} \), and \( B_k = \{ s_i | O_i \cap A_k \neq \phi \} \).

Then, the following facts are easily observed.

**Fact 1.** If \( s_i \in S(q) \), then \( s_i \in B_{w(q)} \).

**Fact 2.** If \( k \in K_{(q)} \), then \( B_k \cap (\cup_{m=1}^{q-1} S(m)) = \phi \).

**Fact 3.** If \( k_1, k_2 \in K_{(q)} \), then the wake-up time of \( s_i \in S(m) \) \((m \neq q)\) remains the same regardless of exchanging \( A_{k_1} \) with \( A_{k_2} \).

Based on the observed facts, the following property, which holds for any optimal schedule, is derived. For the rest of this paper, the superscript \( i \) indicates that it is relevant to \( \pi_i \).

**Lemma 1.** If \( k \in K_{(q)} \), then \( S(q) \subseteq B_k^i \).

**Proof.** Assume that \( s_i \) satisfies \( s_i \in S(q) \), \( s_i \notin B_k^i \) for \( k \in K_{(q)} \). If we exchange \( A_k^i \) with \( A_{w(q)}^i \), the following statements hold true:

1. The schedule length \( l \) does not change.
2. The wake-up time of each sensor node in \( S_{(m)} \) \((m \neq q)\) does not change.
3. The wake-up time of \( s_i \) becomes no larger than \( w(q) - 1 \).

This contradicts the fact that \( \pi_i \) is optimal among the length-\( l \) schedules. This contradiction originates from a wrong assumption. Hence, if \( s_i \in S(q) \), then \( s_i \in B_k^i \) for \( k \in K_{(q)} \). In other words, \( S(q) \subseteq B_k^i \) for \( k \in K_{(q)} \).

To explain Lemma 1 verbally: for a schedule to be optimal, a sensor node should be scheduled consecutively until it goes to sleep from the moment its residual wake-up time becomes the shortest among those of awake sensor nodes. In Fig. 5, we give an example that helps to understand Lemma 1. Fig. 5(a) shows a topology in which there are 4 sensor nodes and 7 objects. For simplicity, each object is assumed to have interference relations with all the other objects. Therefore, the interference graph becomes a fully connected graph. Fig. 5(b) and (c) show the examples of a general schedule and an optimal schedule, respectively. Time slots and corresponding scheduled objects are listed along the x-axis. The y-axis shows the sensor nodes in the order of wake-up time; sensor nodes with a shorter

---

![Figure 5](image-url)

**Fig. 5.** Example for Lemma 1 illustration. In (b) and (c), the time slots and corresponding scheduled objects are listed along the x-axis. The y-axis shows the sensor nodes in the order of wake-up time; the sensor node with a shorter wake-up time is shown in the upper part. A block bordered by a thick line is the one that must be scheduled at that time slot under the given schedule.
wake-up times are shown in the upper part. As a result, it forms a staircase-like shape (by Fact 2). The shaded block indicates that the sensor node is scheduled at that time slot. Under a general schedule, the last block(s) of each stair is bordered by a thick line, indicating that it is the block that must be scheduled at that time slot. This constraint comes from Fact 1. On the other hand, under an optimal schedule, a number of consecutive blocks that stand out should be scheduled according to Lemma 1 as shown in Fig. 5(c). Comparing Fig. 5(b) with Fig. 5(c), it is notable that an optimal schedule reduces the number of unshaded blocks, hence minimizing the total wake-up time of the sensor nodes.

To develop a search algorithm that exploits Lemma 1, we define a permutation function $p: \{1, 2, \ldots, M\} \rightarrow \{1, 2, \ldots, M\}$. Accordingly, $P_M$ is the set of $p$’s. Given that $\pi_1$, $\pi_2 \in \Pi_k$, two schedules $\pi_1$ and $\pi_2$ are said to belong to the same class if $\pi_1(k) = \pi_2(p(k))$, $1 < k < l$ for some $p \in P_c$. When a schedule of length $l$ is given, an algorithm that seeks to find an optimal schedule in the class is described in Algorithm 1. From the initial schedule $\pi(k) = A_k$, each permutation constructs a new schedule $\tilde{\pi}(k) = \tilde{A}_k$. Among the schedules, the optimal schedule is stored in $\pi^*(k) = A_k^*$ after the termination of the algorithm. Note that the search space for permutation is reduced to $N_3!$, which is much smaller than $l!$ of the brute force search.

Algorithm 1: Reduced complexity cost permutation

```
1 begin
2   $J \leftarrow \infty$ // minimum cost
3   for $p \in P_N$, do // $|P_N| = N_3!$
4     $A \leftarrow \{A_1, A_2, \ldots, A_l\}$ // initial schedule
5     $k \leftarrow 1$
6     for $i = 1$ to $N_3$, do // for each sensor node
7         for $A_j \in A$ // for each remaining time slot
8             if $O_p(i) \cap A_j \neq \phi$ then
9                $A_k = A_j$ // construct new schedule
10               $A \leftarrow A - A_j$ // update the remaining set
11               $k \leftarrow k + 1$
12           end
13        end
14    end
15    $J \leftarrow \text{cost}(\tilde{A})$ // calculate new cost
16    if $J < J^*$ then
17       $J^* = J$ // update the optimal cost and schedule
18       $A^* = \tilde{A}$
19    end
20 end
21 end
```

We now present another property of optimal schedules that further reduces the size of the search space. Suppose that time slots up to $w(q)$ are scheduled. In other words, all the neighboring objects of $s_i \in \bigcup_{m=1}^{n} S(m)$ are scheduled already. At this moment, let $\tau_i$ denote the number of remaining time slots of $s_i \notin \bigcup_{m=1}^{n} S(m)$. In addition, $\theta_i$ denotes the number of scheduled time slots of $s_i$ during the time slots in $K(q)$. Then, the following lemma further helps to sift out optimal schedule candidates when performing the permutation.

Lemma 2. If $\tau_i < \frac{1}{|S_q|}(|K_r^q| - \theta_i)$, then $s_i \notin S_{r+1}^q$.

Proof. Assume that $\tau_i < \frac{1}{|S_q|}(|K_r^q| - \theta_i)$ holds for $s_i$, and $s_i$ is chosen as the next scheduling target, i.e., $s_i \in S_{r+1}^q$. Then, the wake-up times of $s_m \in S_{r+1}^q$ and $s_i$ become $w(q_{r+1})$ and $w_i + \tau_i$, respectively. For this schedule to be optimal, the total wake-up time should be smaller than those in any other schedule. Suppose that we reverse the scheduling order of $s_m \in S_{r+1}^q$ and $s_i$ and the new schedule $w_i(w_i + \tau_i)$ increases to $w(q_{r+1}) + \tau_i + |K_r^q|$. Note that the wake-up times of sensor nodes in $\bigcup_{m=1}^{r+1} S(m)$ are not affected by this switching of the scheduling order. Then, the difference between the total wake-up times under the original schedule $w(\pi_i)$ and the new schedule $w(\pi_i)$ is calculated as follows:

$$w(\pi_i) - w(\pi_i) = \left\{ |S_{r+1}^q| w(q_{r+1}) + w_i + \tau_i \right\}$$

$$- \left\{ |S_{r+1}^q| \left( \left( w(q_{r+1}) + \tau_i + |K_r^q| \right) \right) \right\}$$

$$+ w_i(w_i + \tau_i) + \theta_i + \tau_i$$

$$= |K_r^q| - \theta_i - \left| S_q \right| \tau_i$$

$$> 0.$$
We obtain (11) as a result of applying \( w(q) = w(q-1) + \left| K^p_q \right| \) to (10). Moreover, by exploiting the fact that \( \tau_i < \frac{1}{S_i} \left( |K^p_q| - \theta_i \right) \), it is concluded that \( w(\pi^*_i) > w(\pi_i) \).

This implies that \( \pi^*_i \) is not optimal and thus contradicts the assumption. Therefore, \( s_i \neq S^*_{(q-1)} \) if \( \tau_i < \frac{1}{S_i} \left( |K^p_q| - \theta_i \right) \). □

Fig. 6 illustrates how Lemma 2 filters the pool of candidates for the optimal schedule. As in Fig. 5, there are 7 objects to be scheduled. When the MFP ends, the schedule node that has the minimum number of remaining objects first. Thus, in Algorithm 2, the MFP first selects a sensor node with a small number of neighboring objects. The solid and dotted lines indicate the transmission and interference ranges of each object, respectively.

5. Proposed heuristic algorithm

Even though two properties of optimal schedules in the previous section contribute to reducing the size of the search space, it may still be impractical to check all the cases if the number of deployed sensor nodes is not small. To tackle this problem, we present a heuristic algorithm, named the minimum first permutation (MFP), which shows a remarkable performance.

We note that, in general, it is beneficial to schedule the sensor nodes with a small number of neighboring objects first. Thus, in Algorithm 2, the MFP first selects a sensor node that has the minimum number of remaining objects to be scheduled. When the MFP ends, the schedule \( \pi(k) = A_k \) is obtained from the initial schedule \( \pi(0) = A_0 \).

The complexity of the MFP is calculated as follows. The main iteration of lines 5 through 12 is repeated \( N_S \) times with \( |S| \) decreasing from \( N_S \) to 1. Let \( \sigma_i \) denote the number of time slots that satisfy line 8 at the \( i \)-th iteration. Then, the number of operations to check whether the update of \( \tau_m \) is needed in line 12 at the \( i \)-th iteration becomes \( \sigma_i (N_S - i + 1) \). Recalling that \( \sum_{i=1}^{N_S} \sigma_i = l \) (i.e., schedule length) by definition, the total complexity becomes \( O(N_S l) \).

6. Simulation results

In this section, we evaluate the performance of the MFP. For comparison with an optimal solution, we resort to an optimization solver called MOSEK [26]. However, as the problem size grows, the running time of the solver becomes too long to get the result. Therefore, we find a near-optimal solution using a simulated annealing algorithm that runs within a reasonable time.

6.1. General simulation settings

We assume that sensor nodes are deployed in a grid topology as shown in Fig. 7. The size of each grid is \( 10 \times 10 \) m. Each sensor is represented in the figure as a circle, and is located at the center of a grid. The objects, represented as rectangles, are deployed uniformly in the field. The transmission range of an object is expressed as a solid line around the object, and the radius is fixed at 15 m. On the contrary, the interference range, represented as a dotted line around the object, varies depending on the simulation scenario. Even though we fix the transmission range and vary the interference range in the simulation, it is also valid to fix the interference range and vary the transmission range.

**Algorithm 2: Minimum first permutation (MFP)**

1. begin
2. \( S \leftarrow \{s_1, s_2, \ldots, s_{N_S}\} \) // sensor nodes to be scheduled
3. \( A \leftarrow \{A_1, A_2, \ldots, A_l\} \) // initial schedule
4. \( \tau_i \leftarrow |O_i| \) for each \( s_i \in S \)
5. while \( S \neq \emptyset \) do
6. select \( s_i \) with minimum \( \tau_i \)
7. for \( A_j \in A \) do
8. if \( O_i \cap A_j \neq \emptyset \) then
9. \( A_k \leftarrow A_j \)
10. \( A \leftarrow A - A_j \)
11. \( k \leftarrow k + 1 \)
12. // decrease the number of remaining objects for each scheduled sensor node
13. end
14. end
15. \( S \leftarrow S - s_i \)
16. end
17. end

![Fig. 7. A 5 by 5 grid field. Each sensor node is represented as a circle with an index in it, and located at the center of a grid. A rectangle stands for an object. The solid and dotted lines indicate the transmission and interference ranges of each object, respectively.](image-url)
is the set of sensor nodes that are located in the transmission range of $o_1$. Likewise, $S'_1 = \{S_{11}, S_{12}, S_{13}, S_{16}, S_{17}, S_{18}, S_{21}, S_{22}, S_{23}\}$ corresponds to the interference range of $o_1$. At the $o_1$’s scheduled time slot, any object that can interfere with the reception of $o_1$’s transmission should not be scheduled. In the figure, $o_2$ is one of these objects that could cause interference because $S_1 \cap S'_2 = \{S_{22}\}$, whereas $o_1$ and $o_3$ can be scheduled during the same time slot because $S_1 \cap S'_3 = \emptyset$ and $S_1 \cap S'_1 = \emptyset$.

6.2. Simulated annealing

Simulated annealing is an optimization algorithm that is useful especially when the search space is large and discrete. The theory of simulated annealing has its origin in crystal formation from liquids. The concept of the algorithm is analogous to physical annealing, where the material is first heated to a high temperature to permit atomic re-arrangement. Then, a slow cooling process follows to make the material more ordered. As the cooling process advances, the material approaches a stable state, such as a crystal lattice. However, it is also known that cooling to a low temperature itself is not a sufficient condition to find the ground state of the material; if the material is cooled in an uncontrolled manner, it may form a crystal with defects. To obtain a defect-free crystal, careful and slow cooling is crucial.

Starting from a poor and naive solution, simulated annealing tries to find a globally optimal solution as the simulation proceeds. Fig. 8 illustrates the algorithm through the “balls and hills” diagram [27]. In the diagram, the cost is defined as a function of configuration, and the goal is to minimize the cost. In a normal gradient search algorithm, the current configuration is perturbed only in the direction of a downhill move that reduces the cost. However, this approach has a risk of trapping the ball in a local minimum.

To solve this problem, simulated annealing allows some uphill moves that are acceptable under the current temperature. As in physical annealing, the temperature of simulated annealing determines how stringent is the restriction imposed on the cost increase. When the temperature is high, it is relatively easy for the ball to move uphill. As the temperature goes down, the algorithm hardly allows uphill moves, making the ball adhere to a low cost. It is known that the probability of finding a globally optimal solution using simulated annealing with a carefully controlled cooling process approaches one as the annealing schedule is extended [27].

Algorithm 3: Simulated annealing algorithm

```
1 begin
2 $J \leftarrow \infty$
3 $T_0 \leftarrow \log_2(1 + R_{\text{MAX}})$ \quad // initial temperature
4 $R \leftarrow 1$ \quad // current round
5 $A \leftarrow \{A_1, A_2, \ldots, A_{M_{\text{MAX}}}\}$, where $A_j \leftarrow \{o_j\}, \forall o_j \in O$
6 while $R \leq R_{\text{MAX}}$ do
7 \hspace{1em} $T \leftarrow T_0/\log_2(1 + R)$ \quad // current temperature
8 \hspace{1em} for $\forall o_j \in O$ do
9 \hspace{2em} $\tilde{A} \leftarrow A$ \quad // initialize a new schedule
10 \hspace{2em} if uniform(0, 1) $\leq 0.5$ then
11 \hspace{3em} $k \leftarrow$ choose an available time slot for $o_j$ randomly
12 \hspace{3em} change $o_j$’s time slot to $k$ in $\tilde{A}$
13 \hspace{3em} perm_by_w($\tilde{A}$)
14 \hspace{2em} else
15 \hspace{3em} $k_1 \leftarrow o_j$’s time slot
16 \hspace{3em} $k_2 \leftarrow$ choose a time slot randomly
17 \hspace{3em} swap $\tilde{A}_{k_1}$ with $\tilde{A}_{k_2}$
18 \hspace{3em} perm_by_w($\tilde{A}$)
19 end
20 if cost($\tilde{A}$) $\geq$ cost($A$) then
21 \hspace{1em} $\Delta C \leftarrow$ cost($\tilde{A}$) - cost($A$)
22 \hspace{1em} if uniform(0, 1) $\leq \exp(-\Delta C/T)$ then
23 \hspace{2em} $A \leftarrow \tilde{A}$ \quad // uphill move
24 \hspace{1em} end
25 end
26 \hspace{1em} $A \leftarrow \tilde{A}$
27 end
28 if cost($\tilde{A}$) $<$ cost($A$) then
29 \hspace{1em} // update the optimal cost and schedule
30 \hspace{2em} $J' \leftarrow$ cost($\tilde{A}$)
31 \hspace{2em} $A' \leftarrow \tilde{A}$
32 \hspace{2em} end
33 \hspace{1em} $R \leftarrow R + 1$
34 end
35 end
```

Algorithm 3 shows the detailed process of the simulated annealing algorithm tailored for our optimization problem. The total number of rounds $R_{\text{MAX}}$ is set to 100,000 empirically, and a logarithmic cooling schedule is used in line 7.

![Fig. 8. “Balls and hills” diagram. Unlike a normal gradient search algorithm where only downhill moves are allowed, simulated annealing allows some uphill moves under the current temperature to prevent itself from being trapped in a local minimum.](image-url)
As the temperature decreases to 1, it gets harder to allow an uphill move due to a criterion called the metropolis criterion [28] in line 22. In the algorithm, “uniform (0, 1)” is a random number generation function that returns a value between 0 and 1. In each round, the assigned time slot of each object is perturbed in either of two methods. In the first method (lines 11 through 13), an available time slot is chosen randomly for the object. The available time slot means the one to which the object can be reassigned without violating the interference constraint. After changing the object’s time slot to the chosen one, the “perm_by_w (-)” function shuffles all time slots in such a manner that Lemma 1 is satisfied while maintaining the order in which the sensor nodes go to sleep. This can be thought of as applying Algorithm 1 for a given permutation. In the second method (lines 15 through 18), an object’s time slot is swapped with some other randomly chosen time slot. As in the first method, the perm_by_w (-) function is applied to the swapping result.

It is found that our implementation of the simulated annealing algorithm is able to give an optimal solution in a reasonable time when the problem size is small. For example, when the number of objects varies from 15 to 20 in a 3 by 3 grid, it takes 7–11 s for the simulated annealing to get the optimal solution, whereas the optimization solver requires 168–35554 s. The simulation was conducted on computer with a Pentium 4 2.4 GHz CPU and 2 GB of memory running Linux. However, as the problem size increases, the simulated annealing requires more time to obtain the optimal solution. Therefore, instead of spending too much time in running the simulated annealing to find the optimal solution, we limit the running time of the simulated annealing to 20 min in the following simulations. Nevertheless, we believe that the simulated annealing can approximate the optimal solution, and we use it as a close estimate for the optimal solution.

6.3. Performance of MFP

For performance comparison, we compare two schemes: (1) the coloring-only scheme and (2) the coloring + MFP scheme. As for coloring algorithms, GREEDY [24] and DSATUR [25] are employed. The result obtained by the simulated annealing is also presented to show the effectiveness of the MFP compared to the near-optimal solution.

First, to inspect the effect of transmission and interference ranges, we define the interference factor $\alpha$ as the ratio of the interference range to the transmission range. As $\alpha$ increases, more nearby sensor nodes are interfered by transmission from an object. On the other hand, more transmissions can occur concurrently as $\alpha$ gets closer to 1. Fig. 9 shows the total wake-up time of each scheme as a function of $\alpha$. In the rest of the simulation results, the network size of the sensor nodes is 4 by 4, $\alpha$ is 1.6, and the object density, which is the average number of objects per grid, is 20 unless otherwise specified. Because more time slots are needed to schedule all the objects as $\alpha$ increases, the total wake-up time also increases. However, the MFP, which runs as a post-processing step after coloring, reduces the total wake-up time significantly by re-ordering the time slots. Moreover, the performance is comparable to that obtained by the simulated annealing.

Next, we observe the impact of network size and object density on the performance of each scheme. In Fig. 10, it is observed that when the objects are distributed uniformly in the given area, the performance improvement ratio using the MFP decreases with the network size. In particular, the dependency of the performance on the selected coloring scheme seems to be significant when the network size is large. On the other hand, it is observed that the object density has little impact on the performance because the performance improvement ratio remains about the same regardless of the object density.

Lastly, Fig. 11 depicts the simulation result when the objects are not uniformly distributed. Here, network sizes of 5 × 5 and 7 × 7 are considered. To reflect the density of objects located in a certain area, the hot zone is defined as the area of (0.25 · $X_{\text{max}}$, 0.25 · $Y_{\text{max}}$) ~ (0.75 · $X_{\text{max}}$, 0.75 · $Y_{\text{max}}$), where $X_{\text{max}}$ and $Y_{\text{max}}$ are the dimensions of the field. In Fig. 11, the x-axis represents the concentration factor, which determines the object distribution as follows. When a simulation scenario is generated, each object is assigned a probability. If this probability is smaller than the specified concentration factor, the object is located uniformly in the hot zone. Otherwise, the location of the object is decided uniformly in the entire field. Hence, as the concentration factor increases, the hot zone is likely to be crowded with more objects. Compared to the case where objects are distributed uniformly in the entire field, i.e., the concentration factor is 0, the performance improvement becomes salient as more objects are concentrated in the hot zone. Unlike in the uniformly distributed cases, the performance gap between our GREEDY + MFP algorithm and the GREEDY algorithm becomes larger as the network size gets larger. A similar tendency was observed from the results of DSATUR and DSATUR + MFP, which we omit in this paper.

6.4. Effect of coloring scheme

In this subsection, we give an example which shows that an optimal coloring does not necessarily result in an
optimal schedule. Suppose that there are 13 sensor nodes and 4 objects. For simplicity, the transmission range is assumed to be the same as the interference range. The neighbor relations between sensor nodes and objects are $S_1 = \{s_1, s_2, s_3, s_4, s_5, s_6, s_{13}\}, S_2 = \{s_1, s_{12}\}, S_3 = \{s_7, s_8, s_9, s_{10}, s_{11}\},$ and $S_4 = \{s_{13}, s_{11}\}$. It is not difficult to see that the chromatic number of the graph is 2, and one of the corresponding schedules is $p(1) = \{o_1, o_2\}, p(2) = \{o_3, o_4\}$. The total wake-up time of this schedule is 20. On the contrary, if we have a schedule of $p(1) = \{o_1\}, p(2) = \{o_3\}, p(3) = \{o_2, o_4\}$, the total wake-up time becomes 18, which is less than that of the schedule based on the optimal coloring. This example clearly reveals that an optimal schedule may not belong to $\Pi_{\chi(G)}$.

To evaluate the impact of the coloring algorithm on the overall scheduling performance, we use the variants of the two coloring algorithms. The variants produce slightly different coloring results according to several factors. The factors include the way in which the initial vertex set is ordered and how the tie-breaker works when many choices are available during the coloring process. For each scenario, 12 coloring results are obtained using these variants, and a total of 120 scenarios are executed. Among the 12 coloring results for a scenario, $p_0$ denotes the one with the minimum number of colors. For a certain schedule $p$, two points $((l(p)/l(p_0)), w(p)/w(p_0))$ and $(l(p)/l(p'), w(p_{MFP})/w(p_{MFP}))$ are depicted in Fig. 12, where $p_{MFP}$ is the resultant schedule obtained by applying the MFP to $p$.

In Fig. 12, it is observed that employing a better coloring scheme is generally beneficial because the scheduling performance is improved with the coloring performance. However, the points below the dotted line (i.e., the points with $l(p) > l(p')$ and $w(p) < w(p')$) imply that a smaller number of colors does not guarantee a better scheduling performance. It is also observed that applying the MFP is effective in leading the performance ratio of the scheduling to approach 1, as indicated by the two arrows in the figure.

![Fig. 10. The impact of network size and object density on the total wake-up time. When objects are uniformly distributed, the impact of the network size is stronger than that of the object density.](image)

![Fig. 11. The impact of object distribution. The concentration intensity determines what fraction of the objects are located in the hot zone.](image)

![Fig. 12. The impact of the coloring algorithm on the overall scheduling performance. A total of 2880 points are depicted in the figure. They show how much the coloring algorithm and the MFP influence the scheduling performance.](image)
This means that similar scheduling performances can be obtained regardless of the employed coloring scheme. In summary, even though a better coloring algorithm improves the scheduling performance in general, it is acceptable to use a coloring algorithm with reduced complexity especially when the running time is an important metric. The MFP contributes to lessening the impact of the employed coloring scheme.

7. Related work

In a target tracking system, such as invasion monitoring, multiple sensor nodes deployed in an area of interest collaborate to detect the appearance of a target. Once some of the sensor nodes sense a target, they continuously monitor the location of the target. The major concern of this system is to find a good trade-off point between the energy saving of the sensor nodes and the tracking performance, such as the accuracy and coverage. Up to now, much research has been devoted to addressing this problem [6–9]. The considered system in this paper utilizes collaboration between objects and sensor nodes, i.e., message exchange, differently than a tracking system that relies on the relevant target sensing technique. Hence, for the target tracking system, the energy-saving scheduling used in this paper cannot be applied.

A location system such as inventory management and logistics system, keeps track of objects via periodical communications between deployed sensor nodes and objects. It has received much attention over the past few decades, which stimulated the development of various location systems [1–4,29]. A few studies have been done on energy saving for location systems [30–32]. However, to the best of our knowledge, our work is the first that exploits the energy-saving scheduling used in this paper cannot be applied.

Some similar graph coloring problems have been investigated in the literature [33–35]. In [33], Fabri solved a compile-time memory allocation problem using an interval coloring problem. Given a graph \( G = (V, E) \) and positive integral vertex weights \( w : V \rightarrow \mathbb{N} \), the interval coloring problem seeks to find an assignment of an interval \( I(u) \) to each vertex such that two constraints are satisfied: (i) \( \forall u \in V, |I(u)| \leq w(u) \) and (ii) for every pair of adjacent vertices \( u \) and \( v \), \( I(u) \cap I(v) = \emptyset \). The goal of the interval coloring problem is to minimize \( \sum_{u \in V} |I(u)| \). On the other hand, the max-coloring problem seeks to find a proper vertex coloring whose color classes of \( C_1, C_2, \ldots, C_k \) minimize \( \sum_{i=1}^{k} \max_{v \in C_i} w(v) \). It has been applied to many practical problems, such as [36,37]. In [34] and [35], Kubicka and Supowit independently introduced the sum coloring problem. Given a graph, the sum coloring problem seeks to find an assignment of colors, which are positive integers, to each vertex of the graph such that the sum of these integers is minimal. However, the above-mentioned graph coloring problems have different objectives from that of ours, so the algorithms designed to solve these problems cannot be applied to our problem.

8. Conclusion

In this paper, we considered a location system where the positions of multiple objects are tracked by a number of stationary and battery-powered sensor nodes. Unlike previous works, which dealt with the localization algorithm, we aimed at minimizing the total energy consumption of the deployed sensor nodes, thereby prolonging the lifetime of the system.

The goal of minimizing the energy consumption was formulated as an optimization problem where the solution is given as a schedule that assigns a time slot to each object. However, because finding an optimal solution is not straightforward, we approached the problem through two steps. First, an undirected graph equivalent to the problem was constructed. For the graph, a vertex coloring was carried out using a well-known coloring algorithm. Then, the optimal schedule for the given coloring result was obtained by permuting the time slots according to the properties of optimal schedules we presented. However, if the number of sensor nodes is not small, this problem is still intractable. To tackle this, we presented a heuristic algorithm called the MFP, which lowers the complexity of the considered permutation problem significantly.

Through extensive simulations, we evaluated the performance of the MFP. The results confirmed that the total wake-up time of the sensor nodes was improved considerably by applying the MFP to the coloring result. Moreover, the performance was comparable to a near-optimal solution obtained by the simulated annealing algorithm.

Although our algorithm was proposed in the context of location systems, it is also applicable to any problems that can be formulated as the optimization problem dealt with in this paper.

Acknowledgement

This work was partly supported by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2010-(C1090-1011-0004)) and the IT R&D program of MKE/KEIT [K1001809, Intelligent Wireless Communication Systems in 3 Dimensional Environment].

References


Jongwook Lee received B.S. degree in Electrical Engineering from Seoul National University, Seoul, Korea in 2005. He is currently working towards the Unified Master’s and Doctor’s Course in Seoul National University. His current research interests include MAC protocol design, cross-layer optimization and QoS provisioning for wireless sensor network, wireless mesh network and next generation network.

Saewoong Bakh received B.S. and M.S. degrees in Electrical Engineering from Seoul National University in 1984 and 1986, respectively, and the Ph.D. degree from the University of Pennsylvania in 1991. From 1991 through 1994 he was with AT&T Bell Laboratories as a member of technical staff where he worked for AT&T network management. In 1994, he joined the school of electrical engineering at Seoul National University and currently serves as a professor. He has been serving as TPC members for various conferences including ICC, GLOBECOM, INFocom, PIMRC, WCNC, etc. He is on the editorial board of Journal of Communications and Networks (JCN). His areas of interests include performance analysis of communication networks and network security. He is an IEEE senior member and a member of Who’s Who Professional in Science and Engineering.

Jin-Choo Choi received the B.S., M.S., and Ph. D. degrees in the school of Electrical Engineering & Computer Science, Seoul National University in 1998, 2000, and 2005, respectively. From 2006 to 2007, he worked for Samsung Electronics as a senior engineer. In 2009, he has been working as TPC members for various conferences including ICC, GLOBECOM, INFOCOM, PIMRC, WCNC, etc. His research interests include performance analysis of communication networks, packet scheduling policy in wireless networks, and wireless sensor network.