Reliability-Aware Power Adjustment in Air-Soil Wireless Sensor Networks

FANG XIAOLIN¹, GAO HONG¹ AND LI JIANZHONG¹

School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China 150001
E-mail: xlforu@hit.edu.cn
E-mail: honggao@hit.edu.cn
E-mail: lijzh@hit.edu.cn

Received: April 21, 2011. Accepted: June 28, 2011

We study the power conservation and disseminating reliability problem in precision agriculture applications in this paper. Precision agriculture needs to accurately evaluate farming environment, and soil monitoring is an important part of this effort. Soil measurement requires soil sensors being buried deep enough so as to avoid plowing and other mechanical activities. Due to dielectric loss, reflection and many other reasons, radio communication underground is difficult. But existing work find that the communication between aboveground and underground (air-soil) is feasible, thus aboveground sensor nodes are employed to connect the underground nodes. In such air-soil sensor network, most transmissions are dependent on the aboveground sensor nodes, thus, the energy consumption of them is a critical concern. On the other hand, the reliability of data dissemination must also be well considered as a result of important packets forwarding, such as query commands disseminating, control information delivering and binary reprogramming over the air. Given the situation of limited power and unreliable wireless links, we propose the K-matching problem, which is to minimize the energy consumption by setting optimal transmitting power so that each node underground is connected to at least K aboveground nodes in the air. We prove that this problem is NP-complete and two centralized and one distributed approximation algorithm are provided to solve this problem. Theoretical analysis and experiment results demonstrate the effectiveness of the proposed algorithms.

Keywords: ?

1
1 INTRODUCTION

Precision agriculture integrates a suite of technologies that retain the benefits of large-scale mechanization and recognizes farm field variation. It seeks to avoid applying inflexible practices to a crop, and helps to better assess farm situations, such as water content, nutrients, soil temperature, soil moisture, and so on [10]. Therefore, soil sensors are required to monitor the soil environment parameters. Unfortunately, traditional soil measurement using wired sensors must be installed and removed frequently, which involved many human efforts.

The use of Wireless Underground Sensor Networks (WUSN) offers a new development of precision agriculture [21]. In WUSN applications, soil sensors should be buried deep enough so as to keep away from plowing and other mechanical activities. For most farming activities, a burial depth of 40cm is the most secure and balanced option [20]. But radio signal is extremely weak between two nodes underground, therefore, aboveground nodes are still required for data collecting, control information disseminating and reprogramming. M. J. Tiisanen, et al. study the aboveground-underground (air-soil) communication in [25] and the experimental results show that communication between underground and aboveground is feasible. By utilizing a particular wideband discone antenna, the communication range is more than 30m for the burial depths less than 45cm. The experiment of air-soil communication is further conducted in [22] which reports that the communication distance varies in different soil situations.

The architecture of air-soil sensor network is illustrated in Fig. 1. There are many aboveground nodes (AG nodes) and a number of underground nodes (UG nodes) in the network. The UG nodes can rarely exchange information due to bad communication condition underground. The AG nodes are placed aboveground to connect the UG nodes and relay packets. The communication range between AG nodes is much longer that between AG and UG nodes [25, 13], so we don’t consider the communication between AG nodes in this paper.

The users often need to disseminate large amount of data to the sensor nodes in order to response to farming environment variation. For example, query

![Network architecture](image)
commands[6], control information[17] and wireless reprogramming binaries [14] need to be transmitted to the UG nodes through the AG nodes. The AG nodes play an important role of bridging the base station and the UG nodes, and they will consume much more energy. Therefore, energy conservation of these AG nodes should be considered.

We try to reduce the energy consumption of the AG nodes by adjusting the transmitting power in this paper. Transmission at high level power consumes more energy [1], thus decreases the lifetime of sensor nodes. AG node $s$ should set its transmitting power at a proper level. As shown in Fig. 2, if the AG node $s$ wants to transmit data to all the four UG nodes, the proper transmitting power of $s$ should be set to 27. If $s$ wants to send data to only $t_1$ and $t_2$, the proper transmitting power of $s$ should be set to 11 which will be enough.

Reliability is another critical concern in air-soil sensor networks. A. R. Silva, et al. find that the communication quality varies significantly in different conditions, such as different burial depth, soil moisture and so on[22]. Failures in data dissemination may lead to unacceptable or even dangerous consequences. For example, control information must be correctly delivered or else the network functionality is greatly different. And reprogramming binary should be send absolutely correctly, or else the reprogrammed sensor nodes will die.

Matching each UG node with multiple AG nodes is an effective way to improve the reliability of receiving disseminated data for UG nodes. We say that a UG node is matched with a AG node if it can receive data from this AG node. If a UG node is $K$-matched, it can receive disseminated messages even if $K - 1$ of its matched AG nodes fail.

In order to reduce the energy consumption and improve the transmitting reliability, we propose the $K$-matching problem in this paper. The $K$-matching problem is summarized as matching each UG node with $K$ AG nodes, so as to minimize the sum of the proper transmitting power for the selected AG nodes. To our best knowledge, this is the first work that consider both energy conservation and reliability in air-soil sensor network.

In data collecting case, setting the transmitting power of UG nodes and matching them with $K$ AG nodes to reduce energy consumption of UG nodes and improve the reliability of data collecting, is essentially the same as the $K$-matching problem. Therefore, we only consider the data dissemination case in this paper.

![Diagram of different transmitting cost](image_url)
The contributions of this paper are as follows.

- We consider both the energy conservation of the AG nodes and the reliability of data dissemination in air-soil network of precision agriculture applications.
- We present the K-matching problem. Two centralized algorithms and one distributed algorithm are provided to solve this problem. Approximation ratios are also provided for all the three algorithms. Experiments are conducted to confirm our theoretical results.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 presents the definition of the K-matching problem. Section 4 proposes two centralized algorithms for the problem, and a distributed algorithm is presented in Section 5. Performance evaluations are shown in Section 6 and Section 7 concludes this paper.

2 RELATED WORKS

The use of Wireless Sensor Networks has the potential to impact precision agriculture[21, 24, 5]. Due to the restriction of limited battery power, energy conservation is a critical issue in wireless sensor networks. Transmission with excess power will reduce the lifetime of sensor nodes, so power management should be well considered.

There are many works on power management in wireless sensor network. [4, 7, 27, 19] study a common transmitting power problem where every sensor nodes set a common transmission range throughout the whole network. In most applications, the transmission power can not be minimized just by using a common transmitting power. An effective method by assigning each node with a proper transmitting power can greatly improve the lifetime of sensor network. J. Gomez, et. al. [11] investigates the need of variable transmitting power control in wireless ad hoc and sensor networks.

Studies of [26, 15, 8] try to find solutions to control transmitting power so as to reduce packet collisions and increase battery life. The proper adjustment of transmitting power in dynamic network where fast, reliable and real time methods should be provided is studied in [30, 3, 16].

In our paper, we don’t focus on power assignment of all sensor nodes but only on the AG nodes. UG nodes only collect sensing data and rarely forward packets. But the AG nodes are responsible for both jobs, so they will consume more energy and should be paid more concern.

For each UG sensor node, we match it with multiple AG nodes to increase reliability and robustness of data dissemination. Matching each UG node with multiple AG nodes makes our problem seem like the set K-cover problem[23]. But the two problems are quite different. In the set K-cover problem, the cost of a set is a constant value while in our problem the cost of a AG nodes is
adjustable. We treat the transmitting power that a AG node is set as the cost of this AG node later in this paper.

We briefly introduce the Set K-cover problem. Set K-cover problem [23] is to partition all the sensor nodes into mutually exclusive sets such that each set of sensor nodes can cover the whole monitoring area. The problem in [9] is to find the minimum sensors such that the whole monitoring area is covered \( K \) times.

Z. Abrams, et. al. [2] relaxes the restriction that the whole area should be K-covered. The authors give a situation where there may be some area that can not be covered by \( K \) sensors. The goal of [2] is to partition the sensors into \( K \) sets such that the total covered area of each set is as most as possible.

A method to find the maximum \( K \) is proposed in [12], but it doesn’t give a way to partition the sensors into different sets. When maximum set number \( K \) is known, how to divide nodes into each set according to known set number \( K \) is well studied in [29].

Previous work gives solution to K-cover the monitored field, and assumes a disk communication model which is not practical. They try to solve the problem by offering heuristic or randomized algorithms but no ratio bound is provided. In addition, the cost of set is constant in set K-cover problem, but the cost of a AG node is adjustable in our K-matching problem in this paper. One AG node may match more UG nodes if it sets the transmitting power at a higher level, but the energy consumption is also higher. Our objects is to find the minimum total energy consumption of AG nodes, so the AG nodes should be set to a proper power satisfying that every UG node is K-matched.

3 PROBLEM DEFINITION

As introduced above, the communication difficulty underground makes the utilization of abo v eground nodes necessary in wireless sensor network of precision agriculture applications. Given a air-soil sensor network, we are interested in this paper is to find an effective way to reduce the energy consumption in data dissemination. The AG nodes play a significant role in this kind of networks. Due to the restriction of battery life, energy is a critical concern. Each AG node should be set at a proper transmitting power level so as to save energy consumption. Correctness in disseminating data such as query commands and network programming binary must be guaranteed. We match each UG node with \( K \) AG nodes to improve the reliability of transmitting data to the UG sensor nodes. The formalized definition of K-matching problem follows.

Assume a network with \( m \) AG nodes and \( n \) UG nodes. Denote the AG node set as \( S \) and the UG node set as \( T \), where \( |S| = m \) and \( |T| = n \). The candidate underground neighbor of a AG node \( s_i \) is denoted as \( T(s_i) \), e.g. the UG nodes that \( s_i \) can communicate with when the transmitting power of \( s_i \) is set at the
FIGURE 3
Example illustrates 1-matching problem

highest level. Apparently, \(|T(s_i)| \leq n\). On another side, for a UG node \(t_j\), there is a candidate AG node set \(S(t_j)\) that can be used to match it, and \(|S(t_j)| \leq m\). We use \(c(s_i, t_j)\) to represent the cost between AG node \(s_i\) and UG node \(t_j\), e.g. the lowest transmitting power that AG node \(s_i\) needs to be set to match UG node \(t_j\). Take Fig 3 as an example. \(S = \{s_1, s_2\}\), \(T = \{t_1, t_2, t_3, t_4\}\), the cost \(c(s_i, t_j)\) are the numbers beside each link. Clearly, \(T(s_1) = \{t_1, t_2, t_3\}\) and \(T(s_2) = \{t_2, t_3, t_4\}\). On the other side, \(S(t_1) = \{s_1\}\), \(S(t_2) = \{s_1, s_2\}\), \(S(t_3) = \{s_1, s_2\}\) and \(S(t_4) = \{s_2\}\). If we find a matching \(T(s_1)' = \{t_1, t_2\}\), \(T(s_2)' = \{t_3, t_4\}\), then the transmitting power \(s_1\) and \(s_2\) should be set to \(\max_{t \in T(s_1)'} c(s_1, t) = 5\) and \(\max_{t \in T(s_2)'} c(s_2, t) = 15\) respectively.

The objects of this paper is to select a number of AG nodes, and each selected AG node \(s_i\) matches a subset of UG nodes \(T(s_i)'\) from \(T(s_i)\) \((i = 1, 2, \cdots, m)\), so that every UG node is matched with at least \(K\) AG nodes, and the total transmitting power the selected AG nodes are set is minimum. In another word, find a subset of AG nodes \(S(t_j)\) for every UG node \(t_j\) \((j = 1, 2, \cdots, n)\) such that \(|S(t_j)| \geq K\) while minimizing the total transmitting power that the selected AG nodes are set. Our problem can be formally described as finding a K-matching so that

\[
\min \sum_{i=1}^{m} \max_{t \in T(s_i)'} c(s_i, t)
\]

satisfying:

\[
|S(t_j)| \geq K, j = 1, 2, \cdots, n
\]

We treat the cost of a AG node as the transmitting power that the AG node is set later in this paper. The description of minimum cost K-matching problem is simple, but it’s hard to find an efficient solution. We prove that the decision version of K-matching problem is NP-complete.

**Definition 1.** The decision version of minimum cost K-matching problem (named as KM for brevity) is defined as, whether a K-matching exists satisfying

\[
\sum_{j=1}^{m} \max_{t \in T(s_j)'} c(s_j, t) \leq \eta
\]
Theorem 1. The KM problem is NP-complete.

Proof. First, the KM problem is NP problem. Apparently, a nondeterministic polynomial time algorithm for matching each UG node with \( K \) AG nodes exists, and it can be checked whether the cost is no more than \( \eta \) in polynomial time.

Second, KM problem is NP-hard. An instance of KM problem is obtained in the following two steps. Step 1, define that all the costs between AG nodes and UG nodes are 1, then condition (3) can be expressed as

\[
\sum_{i=1}^{m} \lambda(|T(s_i)'|) \leq \eta
\]

where

\[
\lambda(x) = \begin{cases} 
0 & \text{if } x = 0, \\
1 & \text{if } x \geq 1.
\end{cases}
\]

The meaning is that the cost of the selected AG nodes is 1, and unselected AG nodes’ cost is 0.

Step 2, let \( K = 1 \). The instance is finding a 1-matching for all the UG nodes with no more than \( \eta \) AG nodes.

We will transform the set-cover problem to the instance of KM above. The set-cover problem is: Given an element set \( U = \{u_1, u_2, \cdots, u_n\} \), and a series of subsets \( C = \{C_1, C_2, \cdots, C_m\} \), \( C_i \subseteq U \), can \( U \) be covered by no more than \( K \) subsets in \( C \)?

Let \( \eta = K \). Treat the element set \( U \) as the UG node set \( T \), and \( C_i \) as \( T(s_i) \), where \( i = 1, 2, \cdots, m \). Clearly, a solution for set-cover problem exists if and only if the instance of KM problem has a solution.

Therefore, the KM problem is NP-complete.

\[\square\]

4 CENTRALIZED APPROXIMATION ALGORITHMS

It’s proved that if \( \text{NP} \neq \text{P} \), there is no deterministic algorithm with polynomial-time complex to solve the KM problem. We will give two centralized approximation algorithms in this section. The approximation ratio is also analyzed following the algorithms respectively.

What we want to do is to find a way to K-match each UG node and try to minimize the energy cost of the AG nodes. At this point, the centralized greedy algorithm is presented in Algorithm 1.
As described in Algorithm 1, we assign each UG node with a count whose initial value is $K$. The cost-effective function of a AG node $s_i$ is shown as $e(s_i) = \frac{\max_{t \in T(s_i)} c(s_i,t)}{\text{count}(t)}$. The greedy strategy chooses the minimum cost-effective AG node every iteration. Once a AG node is selected, it match all the UG nodes left unmatched and set itself with the lowest power level that can match all these leftover UG nodes. After that, the count of the corresponding UG nodes decrease 1. If the count of a UG node is decreased to 0, then the UG node is K-matched and should be removed from $T(s)$ of it’s candidate AG nodes. On termination, all UG nodes are K-matched if they can be K-matched. A subset of AG nodes are selected, and each one match a number of UG nodes.

An example is shown in Fig.4. We describe in detail the the process of Algorithm 1 for 1-matching problem. Before starting Algorithm 1, $T(s_1) = \{t_1, t_2, t_3\}$, $T(s_2) = \{t_2, t_3, t_4\}$. Because $\frac{21}{3} < \frac{22}{3}$, so $s_1$ is selected at the first iteration. Then, $s_1$ is matched with $t_1$ and $t_2$, and the count of $t_1$ and $t_2$ is decreased by 1. After that, $s_1$ is matched with $t_3$, and the count of $t_3$ is decreased by 1. Finally, $s_2$ is matched with $t_4$, and the count of $t_4$ is decreased by 1.
iteration and \( s_1 \) set itself with cost 21. After that, \( T(s_1) = \emptyset, T(s_2) = \{t_4\} \).

At the second iteration, \( s_2 \) is selected and is assigned with cost 22. Then Algorithm 1 finishes and all UG nodes are 1-matched. Table 1 shows the iterations of Algorithm 1.

Algorithm 1 iterates at most \( Kn \) times. Each iteration needs at most \( m + \max\{|T(s_d)|\} \cdot \max\{|S(t)|\} \leq m + mn \) computation. So, the computational time complex is \( O(Kmn^2) \).

**Theorem 2.** Algorithm 1 is \((\ln K + \ln n + 1) \cdot \frac{c_{\min}}{c_{\max}} \) factor approximation algorithm, where \( c_{\max} \) and \( c_{\min} \) are the maximum and the minimum cost between AG nodes and UG nodes respectively.

**Proof.** The proof is based on article [28]. Before the proofing of Theorem 2, we will define the operation of multiset first. We treat the operation between multiset and set the same as the operation between multisets. The examples are shown below. Given a multiset \( M = \{1, 1, 2, 2, 2\} \) and a set \( A = \{1, 2\} \), the union operation \( M \cup A = \{1, 1, 2, 2, 2\} \), the intersection operation \( M \cap A = \{1, 2\} \) and \( M - A = \{1, 2, 2\} \).

To show that each UG node is matched with \( K \) AG nodes, we duplicate every UG node \( K \) times to construct a multiset. The obtained multiset is denoted as \( T^K = [t_1, t_2, \ldots, t_1, t_2, \ldots, t_2, \ldots, t_n, t_n, \ldots, t_n] \). Assume that \( h \) AG nodes have been selected and \( nh \) elements in \( T^K \) are left unmatched. Let the selected AG nodes be \( S_h = \{s_1, s_2, \ldots, s_h\} \), and the UG nodes they match be \( T'_h = \{T(s_1)' \cup T(s_2)', \ldots, T(s_h)' \} \) in Algorithm 1. In an optimal solution \( OPT \), let \( opt \) be the optimal cost, then the AG nodes can match all the leftover elements at a cost at most \( opt \). Therefore, there must be at least one AG node unselected who contains a cost-effective function no more than \( \frac{opt}{nh} \). Otherwise, if all the cost-effective functions that the unselected AG nodes contain are more than \( \frac{opt}{nh} \), then the optimal solution does not exist, which contradicts with that \( OPT \) is a optimal solution.

Given a AG node \( s \), let the remaining UG nodes that could match \( s \) be \( T(s)' \). The cost-effective function of \( s \) in Algorithm 1 is \( e(s) = \max_{t \in T(s)'} \frac{c(s, t)}{|T(s)'|} \). Assume the minimum cost-effective function that \( s \) contains is \( e(s)_{\min} = \max_{t \in T(s)'} \frac{c(s, t)}{|T(s)'|} \), where \( T(s)' \subseteq T(s)'' \). We can find that \( e(s) \leq e(s)_{\min} \cdot \frac{c_{\max}}{c_{\min}} \). Let \( emin \) be the cost-effective function selected at this time in algorithm 1, then \( emin \leq e(s) \leq
\( e(s)_{\text{min}} \cdot \frac{c_{\text{max}}}{c_{\text{min}}} \) Take Fig.4 as an example. \( c_{\text{max}} = 22 \) and \( c_{\text{min}} = 8 \). The cost-effective function calculated in algorithm 1 is \( e(s_1) = \frac{21}{3} \) and \( e(s_2) = \frac{22}{3} \). Because \( e(s_1) < e(s_2) \), Algorithm 1 selects \( e_{\text{min}} = e(s_1) \) at this iteration. But the minimum cost-effective function is contained in \( s_2 \), it is \( e(s_2)_{\text{min}} = \frac{11}{2} \). The inequations \( e_{\text{min}} = e(s_1) \leq e(s_2) \leq e(s_2)_{\text{min}} \cdot \frac{c_{\text{max}}}{c_{\text{min}}} \) hold.

Let the AG node selected by Algorithm 1 next iteration be \( s_{ih+1} \) next iteration. Assume that the minimum cost-effective function is contained in AG node \( s \). As described above, \( e(s)_{\text{min}} \leq \frac{m}{nh} \), then we get the following results.

\[
e(s_{ih+1}) = \frac{\max_{t \in T(s_{ih+1})} c(s_{ih+1}, t)}{|T(s_{ih+1})|} \leq e(s) \leq e(s)_{\text{min}} \cdot \frac{c_{\text{max}}}{c_{\text{min}}} \leq \frac{\text{opt}}{nh} \cdot \frac{c_{\text{max}}}{c_{\text{min}}} \tag{7}
\]

where \( T(s_{ih+1})^o = T(s_{ih+1}) \cap (T^K - \bigcup_{j=1}^{ih} T(s_j)) \).

Let \( \text{cost}(s_{ih+1}) = \max_{c \in T(s_{ih+1})^o} c(s_{ih+1}, t) \) and \( y_{ij} = |T(s_{ih+1})^o| \), we have

\[
\text{cost}(s_{ih+1}) \leq \text{opt} \cdot \frac{c_{\text{max}}}{c_{\text{min}}} \cdot \frac{y_{ij}}{nh} \tag{8}
\]

For brevity, let \( \alpha = \frac{c_{\text{max}}}{c_{\text{min}}} \). Assume that algorithm 1 finally selects \( f \) AG nodes, we have

\[
\text{cost} = \sum_{j=1}^{f} \text{cost}(s_j) \tag{11}
\]

\[
\frac{\text{cost}}{\text{opt}} \leq \alpha \sum_{j=1}^{f} \frac{y_{ij}}{n_{j-1}} \tag{12}
\]

\[
\leq \alpha \sum_{j=1}^{f} \left( \frac{1}{n_{j-1}} + \frac{1}{n_{j-1} - 1} + \cdots + \frac{1}{n_{j-1} - y_{ij} + 1} \right) \tag{13}
\]

\[
\leq \alpha \left( \frac{1}{Kn} + \frac{1}{Kn - 1} + \cdots + \frac{1}{1} \right) \tag{14}
\]

\[
\leq \alpha (\ln Kn + 1) \tag{15}
\]

\[
\leq \alpha (\ln K + \ln n + 1) \tag{16}
\]

Therefore, Algorithm 1 is \((\ln K + \ln n + 1) \cdot \frac{c_{\text{max}}}{c_{\text{min}}} \) factor approximation algorithm.
Algorithm 2 which is a better choice in minimizing the average cost of all AG nodes. If UG node \( t_j \) needs to be matched with \( n_j \) AG nodes, then the cost is 8. Now four UG nodes are matched by different initial values.

The AG nodes that Algorithm 1 selects always match all the left UG nodes. If the cost between AG nodes and UG nodes is the same for all AG-UG node pair, Algorithm 1 actually chooses the AG node that can match a maximum number of UG nodes every iteration. This algorithm is effective in the situation that less AG nodes should be selected.

The problem that each UG node is matched with different number of AG nodes (we call it any-matching problem), is also NP-complete. Algorithm 1 can be used to find a solution with slight modification by assigning each UG node a count with different initial values.

**Corollary 1.** If UG node \( t_j \) needs to be matched with \( n_j \) AG nodes, then the modified version of algorithm 1 is \((\ln n + n + 1): \frac{\ln n}{\ln 2}\) factor approximation algorithm.

The proofing process is the same as proofing theorem 2, and it is omitted.

Algorithm 1 gives a solution in minimizing the number of selected AG nodes while try to lower the total cost, but it doesn’t offer a good solution to balance the energy cost in the selected AG node. In many applications, hot energy consumption points should be avoid, so the average cost of the AG nodes assigned should be considered.

Next, we give a \((\ln K + n + 1): \frac{n}{\ln 2}\) factor approximation algorithm in Algorithm 2 which is a better choice in minimizing the the average cost of all the AG nodes.

In Algorithm 2, we denote \( L(s, t) \) as the UG nodes that have no more cost than \( c(s, t) \) for AG node \( s \). That is, \( t' \in L(s, t) \iff t' \in T(s) \) and \( c(s, t') \leq c(s, t) \). Take Fig.4 as example, at beginning, \( L(s_1, t_1) = \{t_1\} \), \( L(s_1, t_2) = \{t_1, t_2\} \) and \( L(s_1, t_3) = \{t_1, t_2, t_3\} \). Let the cost-effective function \( e(s, t) = \frac{c(s, t)}{\ln 2} \). Remember that \( T(s) \) may change each iteration, and so does \( L(s, t) \).

Let’s take Fig.4 as example again to illustrate Algorithm 2. Assume that we want to find a 1-matching solution with Algorithm 2. At the very beginning, \( \text{cost}(s_1) = 0, \text{cost}(s_2) = 0, T(s_1) = \{t_1, t_2, t_3\}, T(s_2) = \{t_2, t_3, t_4\} \). At the first iteration, the algorithm select \( s_2 \) because it contains the minimum cost-effective function \( \frac{1}{3} \). \( s_2 \) is selected and sets itself with cost 11. Now \( t_2 \) and \( t_3 \) are matched by \( s_2 \), thus \( T(s_1) = \{t_1\} \). Although \( t_2 \) and \( t_3 \) are matched by \( s_2 \), they should not removed from \( T(s_2) \). This is because they will be used to calculate the cost-effective function of \( s_2 \) next iteration. So \( T(s_2) = \{t_2, t_3, t_4\} \). At this time, \( \text{cost}(s_1) = 0 \) and \( \text{cost}(s_2) = 11 \). At the second iteration, \( s_2 \) is selected again because \( \frac{1}{3} < \frac{3}{7} \) and sets itself with cost 22. Now \( t_1, t_2 \) and \( t_3 \) are matched by \( s_2 \). At this time, \( T(s_1) = \{t_1\}, T(s_2) = \{t_2, t_3, t_4\}, \text{cost}(s_1) = 0 \) and \( \text{cost}(s_2) = 22 \). At the third iteration, \( s_1 \) is selected and sets itself with cost 8. Now four UG nodes \( t_1, t_2, t_3 \) and \( t_4 \) are all 1-matched. At this time \( T(s_1) = \{t_1\}, T(s_2) = \{t_2, t_3, t_4\}, \text{cost}(s_1) = 8 \) and \( \text{cost}(s_2) = 22 \). The iterations of Algorithm 2 is presented in Table 2.
Algorithm 2: Centralized Greedy Algorithm

Input: \( S = \{s_1, s_2, \cdots, s_n\} \)
\[ T = \{t_1, t_2, \cdots, t_m\} \]
\( T(s_i), c(s_i, t) \) for all \( t \in T(s_i) \), where \( i = 1, 2, \cdots, m \)
\( S(t), j = 1, 2, \cdots, n \)
Output: Assign each AG node \( s \) with a cost so that any \( t \in T \) is K-matched.

1: \( T = \emptyset \); Cost = \( \emptyset \); count\( (t) = K, j = 1, 2, \cdots, n \)
2: for i from 1 to m do
3: \( \text{cost}(s_i) = 0 \)
4: for t in \( T(s_i) \) do
5: flag\( (s_i, t) = 0 \)
6: while \( T \neq \emptyset \) do
7: \( \text{emin} = \infty; d = -1; p = t_i \)
8: for i from 1 to m do
9: \( \text{for } t \in T(s_i) \text{ and } c(s_i, t) > \text{cost}(s_i) \text{ do} \)
10: \( \text{get } L(s_i, t) \text{ from } T(s_i) \)
11: \( e(s_i, t) = \frac{c(s_i, t)}{\text{cost}(s_i)} \)
12: \( \text{if } \text{emin} > e(s_i, t) \text{ then} \)
13: \( \text{emin} = e(s_i, t); d = i; p = t \)
14: \( \text{if } d == -1 \text{ then} \)
15: break
16: \( \text{for } t \in L(s_i, p) \text{ do} \)
17: \( \text{flag}(s_i, t) = 1 \)
18: \( \text{cost}(s_i) = c(s_i, p) \)
19: \( \text{for } t \in L(s_i, p) \text{ do} \)
20: \( \text{count}(t) = \text{count}(t) - 1 \)
21: \( \text{if } \text{count}(t) \leq 0 \text{ then} \)
22: \( \text{for } \forall s_i \in S(t) \text{ do} \)
23: \( \text{if } \text{flag}(s_i, t) == 0 \text{ then} \)
24: \( T(s_i) = T(s_i) - t \)
25: for i from 1 to m do
26: \( \text{Cost} = \text{Cost} \cup \{\text{cost}(s_i)\} \)
27: return Cost

<table>
<thead>
<tr>
<th>iteration</th>
<th>( T(s_1) )</th>
<th>( T(s_2) )</th>
<th>( \text{cost}(s_1) )</th>
<th>( \text{cost}(s_2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{t_1, t_2, t_3}</td>
<td>{t_2, t_3, t_4}</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>{t_1}</td>
<td>{t_2, t_3, t_4}</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>{t_1}</td>
<td>{t_2, t_3, t_4}</td>
<td>0</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>{t_1}</td>
<td>{t_2, t_3, t_4}</td>
<td>8</td>
<td>22</td>
</tr>
</tbody>
</table>

TABLE 2
Iterations of Algorithm 2

Theorem 3. Algorithm 2 is \((\ln K + \ln n + 1) \cdot \frac{\text{cmax}}{\text{cmin}}\) factor approximation algorithm.

Proof. Assume that \( h \) AG nodes have been selected and \( n_h \) elements in \( T^K \) are left unmatched. Let the AG nodes be \( S_h = \{s_{1h}, s_{2h}, \cdots, s_{nh}\} \), and the UG nodes they match be \( T_h = \{T(s_{1h}), T(s_{2h}), \cdots, T(s_{nh})\} \). In an optimal solution OPT, let \( opt \) be the optimal cost, then the AG nodes can match all the remaining
elements at a cost at most $opt$. Therefore, there must be at least one AG node who contains a cost-effective function no more than \( \frac{opt}{n_h} \).

Let the AG node selected at next iteration be \( s_{ih+1} \), and the power level that the AG node sets be \( c(s_{ih+1}, t) \), then we have

\[
e(s_{ih+1}, t) = \frac{c(s_{ih+1}, t)}{|L(s_{ih+1}, t)|} \leq \frac{opt}{n_h} \tag{17}
\]

Actually, Algorithm 2 sets a AG node with power level incrementally. One AG node sets itself with a lower power, then it increases its power level until all the UG nodes are K-matched.

Assume that \( s \) is selected in Algorithm 2. Upon termination, \( s \) has set itself with transmitting power \( \omega \) times, and the power levels it sets are \( c_1, c_2, \ldots, c_\omega \) respectively. Assume that at each time \( s \) assigns itself with a transmitting power, there are \( x_1, x_2, \ldots, x_\omega \) elements in \( T^K \) left unmatched, and let \( y_1, y_2, \ldots, y_\omega \) be the accumulated number of elements matched by \( s \) each time. Because \( \frac{c_i}{y_i} \leq \frac{opt}{x_i}, 1 \leq i \leq \omega \), we have

\[
\sum_{i=1}^{\omega} \frac{c_i}{y_i}(y_i - y_{i-1}) \leq \sum_{i=1}^{\omega} \frac{opt}{x_i}(y_i - y_{i-1})
\]

Let \( y_0 = 0 \), because \( \frac{c_i}{y_i} \leq \frac{c_j}{y_j} \) if \( i < j \), we can get

\[
c_\omega = \sum_{i=1}^{\omega} \frac{c_i}{y_\omega}(y_\omega - y_{\omega-1}) \tag{18}
\]

\[
= \sum_{i=1}^{\omega} \frac{c_\omega y_i}{y_\omega c_i} \frac{c_i}{y_i}(y_i - y_{i-1}) \tag{19}
\]

\[
\leq \sum_{i=1}^{\omega} \frac{c_{\max}}{c_{\min}} \frac{c_i}{y_i}(y_i - y_{i-1}) \tag{20}
\]

\[
\leq opt \frac{c_{\max}}{c_{\min}} \sum_{i=1}^{\omega} \frac{y_i - y_{i-1}}{x_i} \tag{21}
\]
Assume that Algorithm 2 finally selects \( f \) AG nodes, then we get the following results.

\[
\text{cost} = \sum_{j=1}^{f} c_{ij}
\]

\[
\frac{\text{cost}}{\text{opt}} \leq \frac{c_{\text{max}}}{c_{\text{min}}} \cdot \sum_{j=1}^{f} \sum_{i=1}^{a_{ij}} \frac{y_{ji} - y_{ji-1}}{x_{ji}}
\]

\[
\leq \frac{c_{\text{max}}}{c_{\text{min}}} \cdot (\ln K + \ln n + 1)
\]

Therefore, Algorithm 2 is \((\ln K + \ln n + 1) \cdot \frac{c_{\text{max}}}{c_{\text{min}}}\) factor approximation algorithm.

Although Algorithm 2 and Algorithm 1 have the same computational time complex which is \( O(Kmn^2) \) and the same approximation ratio, Algorithm 2 has better performance with assigning the AG nodes with a lower average cost. This is because Algorithm 2 always selects AG nodes containing the minimum cost-effective function. A selected AG nodes assign itself with power incrementally. Every iteration, the selected AG is set at a higher transmitting power that makes it match the neighbor UG nodes in a most efficient way. We will discuss the difference between the two algorithms in section 6.

**Corollary 2.** If target \( t_j \) needs to be matched with \( n_j \) sensor nodes, then the modified version of Algorithm 2 is \((\ln \sum_{i=1}^{n} n_i + 1) \cdot \frac{c_{\text{max}}}{c_{\text{min}}}\) factor approximation algorithm.

## 5 DISTRIBUTED IMPLEMENTATION

We give two centralized approximation algorithms in the previous section, but centralized based algorithms don’t work well in large scale wireless sensor networks. Traveling large amount of data to a computational center consumes huge transmission cost. In addition, because of the characteristic of link instability, data loss is a problem.

Unfortunately, a distributed implementation of Algorithm 1 can not be given with only local information of each sensor node. But the distributed implementation of Algorithm 2 is described in Algorithm 3. In the initial process, each AG node broadcasts the information to its neighbor UG nodes. Each time period, all the UG nodes find the minimum cost-effective function and broadcast them to the AG nodes (step 1). When a AG node receives all the cost-effective function from all their neighbor UG nodes, if it finds that it
contains the minimum cost-effective function itself, it labels itself as selected, sets itself with the minimum cost-effective cost and broadcasts to its neighbor UG nodes (step 2). When a UG node receives the selected message, it check whether it can be matched (step 3). If a UG node is k-matched, it needs to tell other UG nodes to update $T(s_i)$ of their candidate AG sensor nodes (step 4 and step 5).

Take Fig.4 as example once more. In the initial process, $t_1, t_2, t_3$ and $t_4$ get \{ $T(s_1)$, $T(s_2)$, $T(s_3)$, $T(s_4)$ \} respectively. When the timer is fired first time, the four UG nodes then broadcast $msg(s_1, t_3, 2\frac{1}{2})$, $msg(s_2, t_3, 1\frac{1}{2})$, $msg(s_2, t_3, 1\frac{1}{2})$ and $msg(s_2, t_3, 1\frac{1}{2})$ respectively. When $s_2$ receive the broadcasted message, it will label itself as selected, and set itself with cost 11. When the UG nodes receive the selected message, $t_2$ and $t_3$ will be removed. After the second time the timer is fired, $t_1$ and $t_4$ will broadcast $msg(s_1, t_1, 8\frac{1}{2})$ and $msg(s_2, t_4, 22\frac{1}{2})$ respectively. Then Both $s_1$ and $s_2$ will label themselves as selected and set themselves with cost 8 and 22 respectively.

Apparently, we can find that the transmission cost of the selected AG node $s$ is at least 1 and at most $2|T(s)| + 1$. Transmission cost of UG node $t$ is at least 1 and at most $K(\sum_{s \in S(t)} |T(s)|) + K + 1$. In the initial process, both AG nodes and UG nodes need to send information to their neighbors, so all nodes have at least 1 transmission cost. An AG node $s$ may need to label itself as selected $|T(s)|$ times and forward remove message $|T(s)|$ times, so the maximum transmission cost is $2|T(s)| + 1$. In the worst case, an UG node $t$ may have to send effective-function $K(\sum_{s \in S(t)} |T(s)|)$ times until all it’s two-step UG neighbors have been K-matched, so the transmission cost of $t$ is at most $K(\sum_{s \in S(t)} |T(s)|) + K + 1$.

Theorem 4. Algorithm 3 is $(\ln K + \ln n + 1) \cdot \frac{c_{\text{max}}}{c_{\text{min}}}$ factor approximation algorithm.

Proof. We prove that Algorithm 3 will return the same result as centralized Algorithm 2 does, then these two algorithms have the same approximation ratio.

We share the idea of article [18], where the authors assume that the communication range is at least twice than the sensing range, that is, two sensor can communicate with each other if they detect a same target. This assumption is not appropriate for our problem, but we can make the AG nodes communicate with each other through the shared UG nodes.

Before the proof, we present the definition of group. Let AG nodes that share a UG node belong to a group. A AG node may belong to multiple groups. We call a AG node belonging to multiple groups a border node, a AG node belonging to one group an intra node.

At each iteration, Algorithm 3 chooses at most one AG node that contains the minimum cost-effective function among each group. The cost-effective
Algorithm 3: Distributed Greedy Algorithm

In the initial process, every AG node $s$ set $cost(s) = 0$, broadcast $T(s)$ and the corresponding cost to their neighbor UG nodes. Set $T(s) = \emptyset$. Assume the network is synchronized. Every UG node start a timer, and the timer is fired periodically.

**Step 1. For each UG node $t$:**
1. $count(t) = K$
2. $cost(s) = 0, \forall s \in S(t)$
3. $flag(s', t') = 0, s' \in S(t), t' \in T(s')$
4. If the timer is fired and $count(t) > 0$ then
   5. Find $s'$ and $t'$ where $s' \in S(t), t' \in T(s')$, satisfying $c(s', t') > cost(s')$, so that $e = \frac{c(s', t')}{\lambda(t, t')}$ is minimum.
6. broadcast $msg(s', t', e)$ to its neighbor AG nodes.

**Step 2. For each AG node $s$:**
1. Let $e_{min}$ be the minimum $\frac{c(s, t)}{\lambda(t, t')}$, where $c(s, t) > cost(s), t \in T(s)$.
2. If $s$ receives $msg(s', t', e)$ from all neighbor UG nodes then
3. If $e_{min}$ is less than any $e$ except $s = s'$ then
4. Set $cost(s) = c(s, t)$
5. broadcast $msg(select(s, c(s, t)))$ to its neighbor UG nodes.

**Step 3. For each UG node $t$:**
1. If $t$ receives $msg(select(s, c))$ then
2. If $t \in T(s)$ then
3. If $c(s, t) > c$ or $count(t) \leq 0$ then
4. Return
5. If $flag(s, t) = 0$ then
6. $count(t) = count(t) - 1$
7. $flag(s, t) = 1$
8. For $t \in T(s)$ do
9. If $c(s, t') \leq c$ then
10. $flag(s, t') = 1$
11. If $count(t) \leq 0$ then
12. Stop the timer.
13. For all $s' \in S(t)$, remove $t$ from $T(s')$ if $flag(s', t) = 0$.
14. Broadcast $msg(remove_t())$ to its neighbor AG nodes.

**Step 4. For each AG node $s$:**
1. If $s$ receives $msg(remove(t))$ then
2. Broadcast $msg(remove(t))$ to its neighbor UG nodes.

**Step 5. For each UG node $t$:**
1. If $t$ receives $msg(remove(t'))$ then
2. For all $s' \in S(t)$, remove $t'$ from $T(s')$ if $flag(s', t') = 0$.

The function $e(s, t) = \frac{c(s, t)}{\lambda(t, t')}$ is non-decreasing, because $c(s, t)$ remains unchanged and $L(s, t)$ decreases.

At each iteration, intra nodes or border nodes are selected to match UG nodes. If an intra node $s$ contains the minimum cost-effective function (contains minimum $e(s, t)$) among all the AG nodes in its group, then it will be selected in the centralized Algorithm 2, and other AG nodes in this group would not be selected before $s$. This is because the cost-effective function is non-decreasing. If any other AG node $s'$ in this group is selected before $s$, the minimum $e(s', t')$ must be less than $e(s, t)$. Because $e(s', t')$ can’t be changed to be smaller, so this can’t be achieved. Border nodes have the same property.
as intra nodes. If a border node contains the minimum $e$ among AG nodes within all groups it belongs to, it will be selected before other AG nodes in these groups.

At each iteration, there must be at least one AG node be selected. After some period, we finally get the same result as centralized Algorithm 2 does. Therefore, Algorithm 3 is \( (\ln K + \ln n + 1) \cdot \frac{\text{cmax}}{\text{cmin}} \) factor approximation algorithm.

\section{Performance Evaluation}

We evaluate the effectiveness of the algorithms proposed above through simulations in this section. The test data is generated as follows. The communication range is set to 40 meters which is similar with the real AG node to UG node communication range in [25]. Communication between UG nodes is denied. UG nodes can receive message from a AG node if they are within the communication range of that AG node. The cost between a AG node and a UG node is set as a function of the distance between them. In our experiments, the cost is \( \lceil \frac{d}{4} \rceil + \lambda \), which spits the cost into ten different levels. This is a strong assumption, because the signal attenuation is not linear to the distance. But we find that linear or exponential function shows a similar results. \( \lambda \) is a parameter represents the least power sensor nodes need so as to transmit message. In this simulation, \( \lambda \) is set at 5.

We simulate the algorithms in two scenarios. In the first scenario, the density of the network is fixed, while the network scale increases. In this scenario, we randomly place 20 sensor nodes with half of them AG nodes in each area of 50*50. Five cases where the area width ranges from 50 to 250 are tested. In the second scenario, the area of the network is fixed while the density of sensor nodes increases. The area of the network is 250*250, and the number of sensor nodes with half AG nodes deployed is from 200 to 1000 stepped by 200. We also consider the influence of different $K$.

Fig.5 and Fig.6 show the simulation results with randomly generated data. Fig.5 is the simulation results in fixed network density scenario and Fig.6 is the results in fixed network area scenario. We use different line styles and markers to represent different results. Solid line, dash line and dot line represent results with $K = 3$, $K = 6$ and $K = 9$ respectively, while triangle marker and cycle marker denote results of Algorithm 1 and Algorithm 2 respectively.

See Fig.5(a) and Fig.6(a), in different $K$, different network density and different network scale cases, Algorithm 1 selects more or less AG nodes than Algorithm 2 does, but a little more in most of the cases. Algorithm 1 may be used in applications that less AG nodes should be powered on. We observe that when $K$ increases, the number of selected AG nodes increases in both algorithms.
FIGURE 5
fixed network density simulation

FIGURE 6
fixed network area simulation
Algorithm 1 has the same approximation ratio with Algorithm 2, nevertheless, Algorithm 2 has better performance in setting a lower total cost of all the selected AG nodes in general cases. Fig.5(b) and Fig.6(b) confirm our analysis. We find that Algorithm 2 is better in both scenarios. The reason is that Algorithm 2 always assigns AG nodes with a lower cost first, and increases its cost if necessary.

If a balance cost assignment is required, Algorithm 2 offers a better choice. Actually, a balanced energy consumption is an important topic in wireless sensor network. If any hot point consume more energy than other points, it will be left uncovered firstly. Fig.5(c) and Fig.6(c) shows that Algorithm 2 greatly reduces the average cost that the AG nodes set.

7 CONCLUSION

The use of wireless sensor networks impacts the development of precision agriculture. The wireless communication underground is much difficult, therefore, it still requires AG nodes for connecting the UG nodes. In this kind of sensor network, the AG nodes will consume more energy than the UG nodes, and the communication quality is unstable. By combining the energy conservation and reliability concern, we propose the K-matching problem, which is to minimize the energy consumption of the AG nodes while satisfying that each UG node is matched with $K$ AG nodes.

We find that the K-matching problem is NP-complete and propose two centralized approximation algorithms. Both two algorithms are $(\ln K + \ln n + 1) \cdot \frac{c_{\text{avg}}}{c_{\text{min}}}$ factor approximation algorithms. One algorithm may select less AG nodes while the other will set a lower average transmitting power. We also present a distributed implementation which will give the same results as the second centralized algorithm. Experiments with generated data are implemented to demonstrate the effectiveness of our algorithms.

ACKNOWLEDGMENT

This work is supported in part by the National Grand Fundamental Research 973 Program of China under grant 2006CB303000, the Key Program of National Science Foundation of China under grant 60533110, the NSFC-RGC of China under grant 60831160525, the Program of New Century Excellent Talents in University under grant NCET-05-0333.

REFERENCES


