algorithms

MDPI

Article

Optimizing Charging Pad Deployment by Applying a Quad-Tree Scheme

Rei-Heng Cheng [1], Chang-Wu Yu [2,*] and Zuo-Li Zhang [3,*]

[1] School of Information Engineering, Xiamen Ocean Vocational College, Xiamen 361100, China; zhengruiheng@xmoc.edu.cn
[2] Department of Computer Science & Information Engineering, Chung Hua University, Hsinchu 30012, Taiwan
[3] School of Intelligent Manufacturing, Wenzhou Polytechnic, Wenzhou 325035, China
* Correspondence: cwyu@chu.edu.tw (C.-W.Y.); zuolizhang@wzpt.edu.cn (Z.-L.Z.)

Abstract: The recent advancement in *wireless power transmission* (WPT) has led to the development of *wireless rechargeable sensor networks* (WRSNs), since this technology provides a means to replenish sensor nodes wirelessly, offering a solution to the energy challenges faced by WSNs. Most of the recent previous work has focused on charging sensor nodes using *wireless charging vehicles* (WCVs) equipped with high-capacity batteries and WPT devices. In these schemes, a vehicle can move close to a sensor node and wirelessly charge it without physical contact. While these schemes can mitigate the energy problem to some extent, they overlook two primary challenges of applied WCVs: off-road navigation and vehicle speed limitations. To overcome these challenges, previous work proposed a new WRSN model equipped with one drone coupled with several pads deployed to charge the drone when it cannot reach the subsequent stop. This wireless charging pad deployment aims to deploy the minimum number of pads so that at least one feasible routing path from the base station can be established for the drone to reach every SN in a given WRSN. The major weakness of previous studies is that they only consider deploying a wireless charging pad at the locations of the wireless sensor nodes. Their schemes are limited and constrained because usually every point in the deployed area can be considered to deploy a pad. Moreover, the deployed pads suggested by these schemes may not be able to meet the connected requirements due to sparse environments. In this work, we introduce a new scheme that utilizes the Quad-Tree concept to address the wireless charging pad deployment problem and reduce the number of deployed pads at the same time. Extensive simulations were conducted to illustrate the merits of the proposed schemes by comparing them with different previous schemes on maps of varying sizes. In the case of large maps, the proposed schemes surpassed all previous works, indicating that our approach is more suitable for large-scale network environments.

Keywords: WRSN; drones; wireless charging; Quad-Tree

Citation: Cheng, R.-H.; Yu, C.-W.; Zhang, Z.-L. Optimizing Charging Pad Deployment by Applying a Quad-Tree Scheme. *Algorithms* **2024**, *17*, 264. https://doi.org/10.3390/a17060264

Academic Editor: Frank Werner

Received: 16 April 2024
Revised: 5 June 2024
Accepted: 12 June 2024
Published: 14 June 2024

1. Introduction

Wireless sensor networks (WSNs) are a kind of wireless network, facilitated by an architecture that allows *sensor nodes* (SNs) to be deployed, with data transmission typically passing through some other intermediate nodes. Deploying WSNs in inaccessible areas enables automatic data collection, making them valuable for monitoring physical resources such as in military applications, medical care, and safety monitoring [1]. Sensor nodes are often deployed outdoors or in hazardous environments, and their battery life is finite. As a result, power loss can lead to node failure and various serious issues, impacting data transmission and potentially causing network paralysis. Additionally, replacing faulty sensor nodes in challenging conditions poses significant challenges due to accessibility constraints and environmental harshness. The difficulty and expense of battery replacement in such environments have made energy management a critical concern for WSNs. The recent

advancement in *wireless power transmission* (WPT) has led to the development of *wireless rechargeable sensor networks* (WRSNs), since this technology provides a mean to replenish sensor nodes wirelessly, offering a solution to the energy challenges faced by WSNs.

Most of the recent previous work has focused on charging sensor nodes using *wireless charging vehicles* (WCVs) equipped with high-capacity batteries and WPT devices. In these schemes, a vehicle can move close to a sensor node and wirelessly charge it without physical contact. These studies have involved utilizing multi-functional vehicles [2], simultaneous charging of multiple sensor nodes [3], designing mobile charging protocols [4], or planning optimal collaborative charging schedules of multiple vehicles to enhance sensor network performance [5].

While these schemes can mitigate the energy problem to some extent, they overlook two primary challenges when applying WCVs for wireless charging: off-road navigation and vehicle speed limitations [6]. In order to improve the mentioned shortcomings, recent studies [6–10] have explored the use of *unmanned aerial vehicles* (UAVs) for charging sensors in WRSNs. UAVs have now found extensive applications in the commercial sector, including surveillance, remote sensing, aerial photography, search and rescue, and on-demand emergency communication. Here a UAV equipped with WPT devices is called a *wireless charging drone* (WCD, or drone in short) [6]. In [8], Su et al. aimed to enhance the total energy of drones while charging energy-limited devices. Li et al. [9] proposed a method to charge WRSN sensor nodes using drones, considering energy consumption from flight distance and striving to improve charging efficiency. Yang et al. [10] investigated drones equipped with large-capacity batteries to deliver energy to sensor nodes.

One advantage of drones is their ability to access and charge more difficult-to-reach sensor nodes. In addition, the flying speed of drones is faster than the speed of vehicles. However, since the battery capacity of drones is limited, the flight distance of the drone is limited when compared to vehicles. To reduce the need for frequent return trips to the base station for drones, the development of wireless charging pads (pads) for automatic drone landing, combined with high-power and efficient wireless power transmission systems, enables drones to be automatically charged within a short timeframe [6,11,12].

To overcome the off-road navigation and travel speed limitations of vehicles, Chen et al. [6] first proposed a new WRSN model equipped with one drone coupled with several pads deployed to charge the drone when it cannot reach the subsequent stop. Their wireless charging pad deployment aims to apply the minimum number of pads so that at least one feasible routing path from the base station can be established for the drone to reach every SN in a given WRSN. The wireless charging pad deployment originally allowed pads to be deployed to any point in the defined area. However, in [6], Chen et al. proposed three algorithms that only use sensor locations as potential deployment positions for the pads. Their simplified pad deployment problem is quite similar to the geometric connected dominating set, which is known to be NP-complete [6]. The major weakness of Chen et al.'s best three schemes, which are based on graph theory, is that they only consider deploying a wireless charging pad at the locations of the wireless sensor nodes. Their schemes are limited and constrained because usually a pad can be deployed at every point in the deployed area. Moreover, the deployed pads suggested by these schemes may not be able to meet the connected requirements due to sparse environments. On the other hand, the remaining scheme in [6], which is based on geometry, is a blind strategy that does not consider the positions of the sensors at all. As a result, it always requires the deployment of more charging pads.

In [13], Chen et al. proposed a novel and adaptive pad deployment scheme that can adapt to arbitrary locations of the base station, arbitrary geographic distributions of sensor nodes, and arbitrary sizes of network areas. However, this scheme requires an appropriate determination of the number of clusters, which leads to a prolonged processing time.

Based on above discussions, this work introduces a new scheme that utilizes the Quad-Tree concept [14,15] to address the wireless charging pad deployment problem and reduce the number of deployed pads. The Quad-Tree concept represents spatial positions in

a hierarchical manner [14,15], allowing for the efficient exploration of suitable locations for pad deployment. By adopting the Quad-Tree concept, the new scheme not only overcomes the major limitation of previous approaches, which only deploy wireless charging pads at limited locations, but also efficiently identifies the appropriate locations from the infinite possible locations for pad deployment.

Extensive simulations were conducted to illustrate the advantages of the proposed schemes by comparing them with different previous schemes on maps with varying sizes. For small maps, the previous schemes either struggled to find suitable pad configurations for certain maps or experienced prolonged processing times. In contrast, the proposed scheme encountered no issues with these small maps. Additionally, in terms of reducing the number of used pads, the proposed scheme slightly outperformed previous works. For medium-sized maps, the proposed schemes significantly outperformed all previous methods in reducing the number of required pads, with a maximum reduction of 12.37% and an average reduction of 9.59%. In the case of large maps, the proposed schemes surpassed all previous works, indicating that our approach is more suitable for large-scale network environments.

Briefly, the contributions of this work are listed below:

(1) This work firstly applies the Quad-Tree concept to develop a new scheme in order to efficiently reduce the number of deployed wireless charging pads for the charging pad deployment problem.
(2) If all spatial positions are expressed by the leaf nodes of a Quad-Tree, the execution time increases significantly. Therefore, it is necessary to analyze the Quad-Tree node splitting conditions to reduce the number of used Quad-Tree nodes. To further reduce the required execution time of the proposed schemes, we also designed some variations with different minimum units (as the splitting conditions in the Quad-Tree) to reduce the total execution time.
(3) We also conducted extensive simulations to demonstrate the merits of the proposed schemes by comparing the proposed schemes with different previous methods on maps of different sizes.

The rest of this work is organized as follows. Section 2 presents the related work. Section 3 introduces the proposed Quad-Tree schemes, followed by Section 4, which presents and discusses the simulation results by comparing the proposed schemes with previous methods. Finally, Section 5 concludes this work.

2. Related Works

2.1. WPT for WSNs

WPT, a technique for transmitting energy over long distances, finds applications in various domains such as portable devices, vehicles, and UAVs [16]. In the context of wireless sensor networks (WSNs), researchers have explored methods for leveraging WPT to directly supply energy to sensor nodes experiencing energy deficits, thus addressing the challenge of limited energy resources [17]. Wireless charging technology facilitates energy transmission to sensors.

The deployment of wireless charging by drones enables continuous energy replenishment for sensors deployed in inaccessible outdoor environments, eliminating the need for maintaining a physical charging infrastructure. For instance, Joo et al. [18] investigated the efficiency enhancement of a system providing wireless power for rail transportation equipment. They analyzed electromagnetic characteristics using finite element methods, including magnetic equivalent resistance, inductance, magnetic coupling rate, and magnetic core loss. Furthermore, they applied the results of magnetic field finite element analysis to equivalent circuit modeling, analyzing voltage transmission ratio and input/output characteristics of the CLLC resonant converter designed for wireless power transmission.

In recent years, the advancement of wireless power transfer (WPT) technology has enabled its use in providing additional energy for wireless sensor networks [2]. Such wireless rechargeable sensor networks can serve as development platforms [19], comprising a

base station (BS), several wireless communication sensor nodes capable of wireless charging, wireless charging vehicles (WCVs), and unmanned aerial vehicles (UAVs) equipped with WPT devices. The base station collects sensor data and offers rapid battery charging services for WCVs or UAVs, which can then wirelessly charge sensor nodes to replenish their energy.

2.2. Charging Approaches

Wireless charging vehicles (WCVs) or mobile chargers (MCs) have the capability to wirelessly charge sensor nodes within the network area, which can be 1D, 2D, or 3D. Charger types may include charging vehicles, unmanned aerial vehicles (UAVs), or a combination of both. Previous research on wireless charging for wireless rechargeable sensor networks (WRSNs) has predominantly focused on utilizing WCVs [20–24] or UAVs [6,11,16,25–28].

Numerous studies have concentrated on charging sensors using WCVs. For instance, Nguyen et al. [20] introduced a novel on-demand charging algorithm named the Fuzzy Q-Charging Algorithm, aiming to enhance the network lifespan by optimizing the time and location for MCs to perform charging tasks. Fuzzy Q-charging employs fuzzy logic to determine the optimal charging energy for sensors and proposes a method to identify the optimal charging time for each charging position. This approach utilizes Q-learning to determine the next charging position to maximize the network lifespan.

Chen et al. [21] proposed a dual-side charging strategy for mobile charging robot (MR) traversal planning, aimed at minimizing the length, energy consumption, and completion time of the MR's traversal path. Based on MR dual-side charging, adjacent sensors on both sides of the designated path can wirelessly charge via the MR and simultaneously transmit sensory data to the MR. The path construction is based on a power map drawn from the remaining power of sensors in the WRSN and the distance between sensors. Simultaneously, a charging strategy with dual-side charging capability is determined.

To minimize network energy consumption, Zhong et al. [22] designed an energy-minimization path construction algorithm based on dual-function vehicles for data collection and wireless charging. Their proposed algorithm constructs a mobile vehicle path with anchors for data collection and charging sensor nodes.

Li et al. [23] explored charging sensor nodes with non-deterministic mobility and proposed the Predicting–Scheduling–Tracking approach to execute charging tasks based on the network mechanism.

Considering severe node failure problems in networks with high charging demands, Li et al. [24] addressed the issue by considering the dynamic energy consumption rate of nodes based on historical statistical data and real-time energy consumption. They proposed two efficient online billing algorithms, PA and INMA, where PA selects the next charging node based on the charging probability of the requesting node, and INMA selects the node that minimizes the energy consumption of other requesting nodes as a charging candidate.

With the advent of wireless charging UAV technology, several studies have explored the utilization of drones for charging sensors. Chen et al. [6] proposed a method employing a single UAV equipped with pads to charge sensors. They introduced a new model for wireless rechargeable sensor networks (WRSNs) incorporating a UAV and multiple pads strategically deployed to facilitate the UAV's flight path. This model effectively addresses charging challenges by overcoming UAV energy limitations.

Yoon [11] developed multiple Minimum Depth Trees (MDTs) for all nodes, considering both drone and sensor node energy. The parent node dynamically controls data transmission to prevent its own energy depletion, ensuring balanced data collection for all nodes and preventing energy drain in hotspot areas.

In another study, Chen et al. [13] investigated the minimal number of pads required in UAV-based WRSNs. They proposed an adaptive pad deployment scheme capable of accommodating diverse base station locations, geographic distributions of sensor nodes, and network area sizes.

Jin et al. [25] tackled the drone scheduling problem to minimize the total charging time for all sensors under energy constraints. They introduced the Drone Scheduling Algorithm (DSA) to optimize UAV scheduling. Additionally, to ensure sustainable WRSN task execution, they developed the Deadline Drone Scheduling Algorithm (DDSA), prioritizing drones charging the highest number of sensors before deadlines.

Wu et al. [26] studied UAV trajectory optimization to maximize energy utilization efficiency. Liang et al. [27] formulated a charging UAV deployment optimization problem, aiming to increase the number of sensor nodes within charging scopes, thus improving network charging efficiency and reducing UAV motion energy consumption.

Addressing 3D WRSNs, Lin et al. [28] devised a spatial discretization scheme to construct a finite set of charging spots for UAVs and a temporal discretization scheme to determine suitable charging durations for each spot.

3. Optimizing the Charging Pad Deployment Problem by Applying the Quad-Tree Scheme

In this section, we propose algorithms to optimize the charging pad deployment problem using the Quad-Tree scheme.

3.1. Problem, Notation and System Model

3.1.1. Network Model

In a WRSN, many rechargeable sensors, $S = \{s_0, s_1, \ldots, s_n\}$, are randomly scattered in an area. These sensors collect specified information in the network and transmit it to the *base station* (BS). When the battery level of the sensor is low, the BS dispatches drones to the sensor location for charging. Considering the limited power of a drone, it may not be able to fly to the sensor location, perform charging tasks, and then fly back to the BS. Therefore, it is necessary to configure some charging pads in the network area, $P = \{p_0, p_1, \ldots, p_m\}$, so that the drone can replenish power on the charging pads to complete the task. In this article, some additional assumptions about the network are as follows:

(1) There is only one BS and one drone in the WRSN.
(2) Sensors are homogeneous, static, and have the same battery capacity.
(3) The BS knows the location of each sensor.
(4) When a sensor's battery level is low, the BS receives a notification and subsequently dispatches the drone for charging. Once the drone completes the charging process, it returns to the BS.
(5) The BS is located in the center of the network, does not move, and has unlimited power. Likewise, the charging pad does not move and has unlimited power.

In this paper, we use a 2D map to illustrate the positions of sensors, charging pads, and base stations. Figure 1 depicts an example network layout. The drones can fly directly from the base station to sensors 3, 7, 8, and 9 for recharging tasks before returning. Alternatively, they can fly to pad 1/2 for a full recharge before proceeding to sensors 3, 4, 6, and 7/sensors 1, 2, 5, and 9 for recharging tasks, and then return to pad 1/2 to be recharged before flying back to the base station. Upon reaching the sensor locations, the drone hovers to charge the sensors.

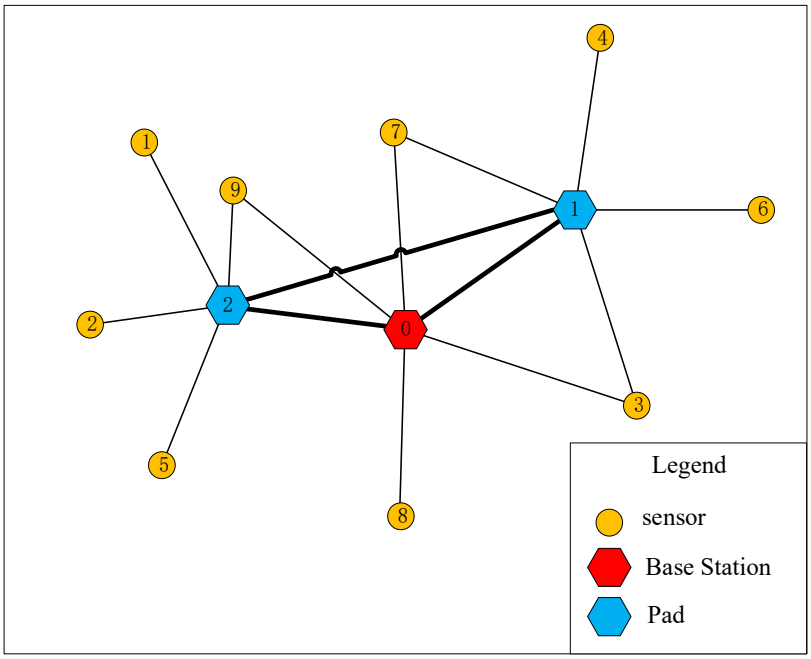


Figure 1. A schematic network layout; the base station is located at the center of the network, and the drone can travel directly or via pads to reach sensor locations for recharging tasks.

3.1.2. Drone Energy Consumption Model

For the convenience of later discussion, we use the symbols listed in Table 1 to represent some characteristics of the drone and sensors.

Table 1. The parameters used to derive the maximum flight distance of the drone.

Symbols	Meaning
E_{sensor}	the maximum battery energy of the sensor (J)
E_{max}	the maximum battery energy of the drone (J)
E_{charge}	the energy required for charging a sensor (J)
P_{fly}	the power consumption of the drone during flight (J/s)
P_{hover}	The power consumption for drone hovers during the charging process (J/s)
P_{speed}	the charging speed (J/s)
T_{charge}	the time to charge a sensor (s)
V_{fly}	the flying speed of the drone (m/s)
$D_{maxcharging}$	The maximum flying distance of the drone for charging tasks (m)
D_{maxpad}	The maximum flying distance of the drone for moving between pads (m)
ρ	the charging efficiency of the drone to the sensor (percentage)

Under the assumption that the drone arrives when the sensor is either at or near depletion of its battery, the drone needs to recharge the sensor with an amount of energy denoted as E_{sensor}. Considering the charging efficiency, the drone can recharge the sensor with a rate of $\rho \times P_{speed}$ energy per second.

Thus, the time required for a drone to complete the charging of a sensor upon reaching its position is as follows:

$$T_{charge} = \frac{E_{sensor}}{\rho \times P_{speed}} \tag{1}$$

Considering that the drone hovers while charging the sensor, the energy required for the charging task needs to account for the energy consumed for hovering:

$$E_{charge} = E_{sensor}/\rho + P_{hover} \times T_{charge} \tag{2}$$

After subtracting the energy used for the recharging task, the drone has a remaining energy of $E_{max} - E_{charge}$ available for flying. Since the drone must return after completing the recharging task, the flying distance must be divided by 2. Therefore, the maximum distance at which the drone can perform recharging tasks can be calculated as follows:

$$D_{maxcharging} = \left(E_{max} - E_{charge}\right) / P_{fly} \times V_{fly} \times \frac{1}{2} \tag{3}$$

On the other hand, when the drone flies between pads, it does not need to reserve energy for recharging tasks or the return trip, as it can be fully recharged at the destination pad. Therefore, the maximum distance between two pads is determined by Equation (4):

$$D_{maxpad} = E_{max} / P_{fly} \times V_{fly} \tag{4}$$

Assuming the sensor remains in the sleeping state while being charged, its energy consumption is not considered. Furthermore, to further simplify the problem, both the charging efficiency and the energy consumption for hovering are disregarded; in other words, $\rho = 1$ and $P_{hover} = 0$.

Thus,

$$D_{maxcharging} = \frac{E_{max} - E_{sensor}}{P_{fly}} \times V_{fly} \times \frac{1}{2} \tag{5}$$

$$D_{maxpad} = \frac{E_{max}}{P_{fly}} \times V_{fly} \tag{6}$$

3.1.3. Problem Definition

In the network model section, the use of pads can extend the drone charging range. Therefore, in a large-area WRSN, the proper configuration of the pads will undoubtedly determine the performance of the network. Based on construction cost considerations, the number of pads should be as small as possible. Since the purpose of using pads is to expand the range of drone sensor charging services, for any sensor X in the network, after building a suitable pad, the drone can find at least one path from the BS to X to complete the charging task. The problem discussed in this article can therefore be defined as follows.

Given a $size \times size$ two-dimension WRSN network N, n sensors $S = \{s_1, s_2, \ldots, s_n\}$ located at $\{(s_{1,x}, s_{1,y}), (s_{2,x}, s_{2,y}), \ldots, (s_{n,x}, s_{n,y})\}$ respectively, a base station B located at $(p_{0,x}, p_{0,y}) = (size/2, size/2)$, and a drone U, the problem is to find out how to place the minimum number m of charging pads $P = \{p_1, p_2, \ldots, p_m\}$ located at $\{(p_{1,x}, p_{1,y}), (p_{2,x}, p_{2,y}) \ldots, (p_{m,x}, p_{m,y})\}$ to satisfy coverage and connectivity constraints as stated in [6].

Let $d(s_i, p_j)$ represent the distance between sensor s_i and pad p_j and let $d(p_i, p_j)$ represent the distance between pad p_i and pad p_j, $P' = \{B\} \cup P = \{p'_0 = B, p'_1, p'_2, \ldots, p'_m\}$, and Ω be a permutation of a subset Q of P'. A two-tuple (x, y) is called a point in two-dimension Euclidean plane R^2; that is, we have (x, y) $\in R^2$. Thus, the objective is to minimize the number of pads located at the two-dimension Euclidean plane R^2, subject to the constraints specified by Equations (8)–(14).

$$\text{Minimize} \sum\nolimits_{\text{every point } (x,y) \text{ in } R^2} \left(\phi_{x,y}\right) \tag{7}$$

Subject to:

$$\phi_{x,y} = \begin{cases} 1 & \textit{if a pad has been deployed at the point } (x, y) \in R^2, \\ & \quad 0 < x \leq size \textit{ and } 0 < y \leq size \\ 0 & \textit{otherwise} \end{cases} \tag{8}$$

$$P = \left\{ p_i \middle| \, p_i = (x_i, y_i) \textit{ and } \phi_{x_i,y_i} = 1, where (x_i, y_i) \in R^2, 0 < x_i \leq size, 0 < y_i \leq size \right\} \tag{9}$$

$$P' = \{B\} \cup P \tag{10}$$

$$\exists!\ \Omega \text{ such that} \sum_{\Omega} \left(a_{0,\Omega(1)} \times \prod_{k=1}^{|Q|-1} a_{\Omega(k),\Omega(k+1)} \times a_{\Omega(|Q|),j} \right) \geq 1,\ \ \forall p_j \in P', \tag{11}$$
$$\text{Where } \Omega \text{ is a permutation of a subset } Q \text{ of } P - \{p_j\}$$

$$a_{i,j} = \begin{cases} 1 & if\ d\left(p'_i, p'_j\right) \leq D_{maxpad},\ \ p'_i \neq p'_j,\ p'_i \ and\ p'_j \in P' \\ 0 & otherwise \end{cases} \tag{12}$$
$$\text{When i } > 0, \text{ j } > 0, \text{ we have } p'_i, p'_j \ni P; \text{ and we have } p'_0 = B$$

$$\sum_{p_j \in P'} b_{i,j} \geq 1,\ \forall s_i \in S \tag{13}$$

$$b_{i,j} = \begin{cases} 1 & if\ d\left(s_i, p'_j\right) \leq D_{maxcharging}, s_i \in S,\ p'_j \in P' \\ 0 & otherwise \end{cases} \tag{14}$$

Where $a_{i,j}$ and $b_{i,j}$, respectively, represent whether the distance between pad i and j is not greater than D_{maxpad}, and whether the distance between sensor i and pad j is not greater than $D_{maxcharging}$.

As for Figure 2, the displayed edges represent the distances that meet the requirements. For example, because the distance between sensor 7 and pad 1 is less than or equal to $D_{maxcharging}$, $b_{7,1}$ is indicated in Figure 2. However, the distance between sensor 8 and pad 1 exceeds $D_{maxcharging}$, so $b_{8,1}$ is not indicated in Figure 2. From a graph perspective, a graph that satisfies the condition of Formula (8) must be a connected graph.

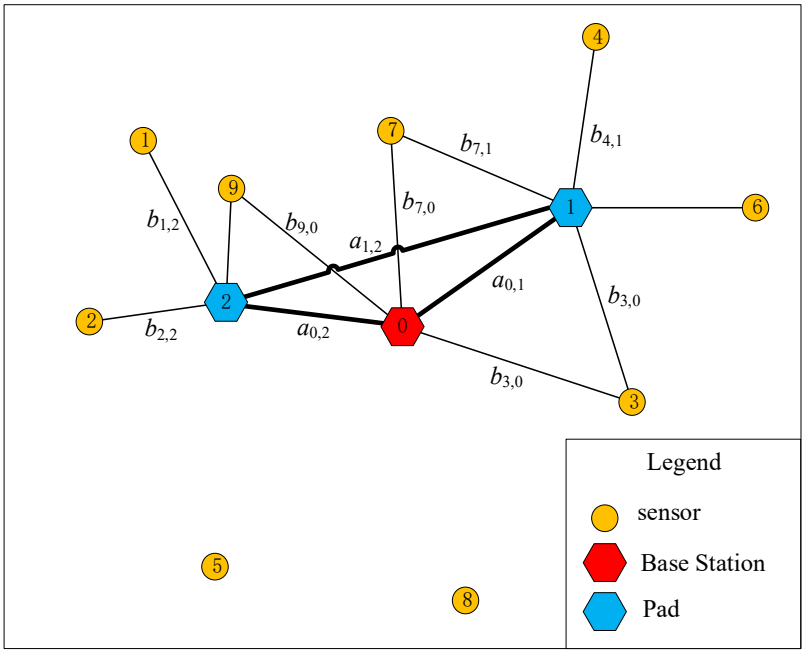


Figure 2. Illustration of parameters a and b. The listed edges represent the distances that satisfy the conditions, indicating feasible drone flight paths.

3.2. Proposed Method

The previous approach only allows pads to be placed where sensors are located, which limits the available placement options. In sparse environments, this approach may fail to meet connectivity requirements. To address these limitations, the proposed approach divides the map into grid points where pads can be placed. However, sensor networks are typically deployed over large areas. For instance, in a 1 km by 1 km square area, dividing it into 1 m by 1 m square grid points would result in 1 million possible locations for deployed pads. Clearly, the solution space becomes too large and complex to find suitable and efficient pad placements. Therefore, this article proposes using the Quad-Tree scheme to represent spatial positions.

The Quad-Tree represents spatial positions in a hierarchical manner. If all spatial positions, such as the grid points mentioned earlier, are represented by the leaf nodes of a Quad-Tree, the total number of Quad-Tree nodes that need to be processed will be much greater than the number of grid points. Therefore, it is necessary to analyze the conditions for splitting Quad-Tree nodes to reduce their number.

In addition to leaf nodes, a Quad-Tree node represents a rectangular area. When information within the subdivided area is needed, the node is divided into four sub-areas. If the information obtained from a Quad-Tree node remains unchanged before and after subdivision, then further subdivision is unnecessary. Consider Figure 3 as an example. Assume the red point represents the center position of Quad-Tree node A, and the blue point represents the center position of its four child nodes B, C, D, and E. If the cover sets (the sets of sensors covered by a pad) resulting from placing a pad at the center position of nodes A to E are all equal, it indicates that Quad-Tree node A does not require further subdivision. This is because placing a pad at the center of node A already covers all sensors in the area.

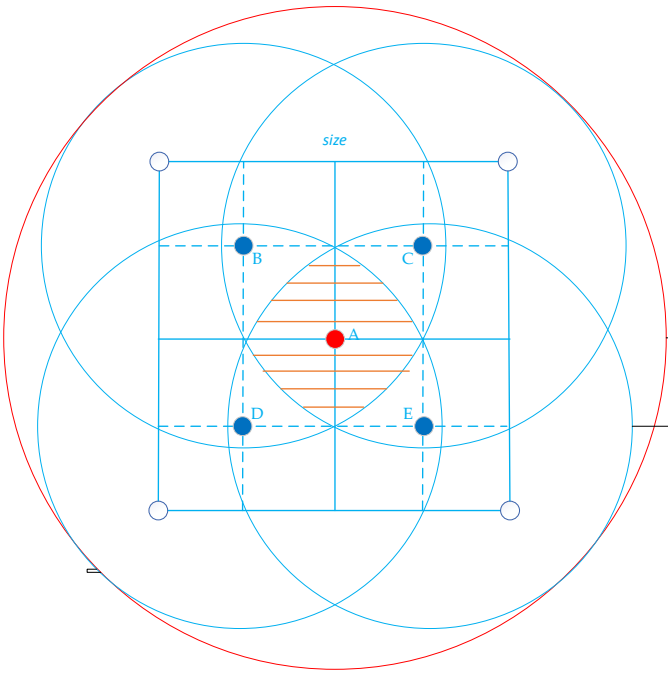


Figure 3. When the Quad-Tree node A covers the same sensors (red circle) as the ones within the range (blue circle) of its four child nodes B, C, D, and E, the Quad-Tree node A does not need to be subdivided into B, C, D, and E.

Therefore, in this article, the content of each Quad-Tree node is designed as listed in Table 2.

Table 2. The information stored in a Quad-Tree node.

Symbol	Meaning
cx, cy	The center position of a Quad-Tree node
size	The side length of the rectangular area represented by a Quad-Tree node
dT	Detailed description will be described later
coverSet	The set of sensors covered by the Quad-Tree node
coverNum	The number of sensors in *coverSet*
status	Whether the Quad-Tree node needs to be split again? 0: Need to be split again. 1: No need to be split again.

In Figure 4 below, all sensors within the radius $D_{maxcharging}$ of the blue dot can be covered by the pad placed at that position (i.e., the UAV can start from the dot position and reach any sensor within that range to charge, and then safely fly back). All sensors within the radius $D_{maxcharging} + dT$ of the center position of the Quad-Tree node (represented by the red dot) include all sensors that may be covered when the pad is placed at any position within this square. The set of these sensors is the *coverSet* of the Quad-Tree node. From Figure 2, we can observe that the value of dT is equal to $\frac{\sqrt{2}}{2}size$.

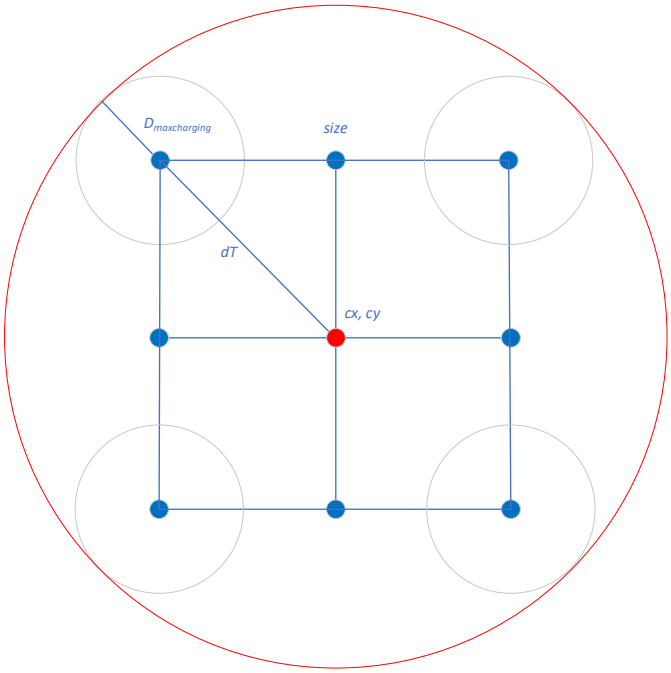


Figure 4. All sensors within the radius $D_{maxcharging} + dT$ (red circle) of the center position of the Quad-Tree node (red dot position) encompass all sensors that may be covered when the pad is placed at any position within this Quad-Tree node's square area.

Assuming that there are n sensors in the map, each Quad-Tree node's *coverSet* needs to be checked every time it is expanded, and the complexity of checking the *coverSet* is O(n). For a map of size m by m, the maximum expanded level L is $\log_2 m + 1$, and the maximum number of nodes that need to be processed is $\frac{4^L-1}{4-1}$, which is proportional to m^2. Therefore, the worst complexity of Quad-Tree construction (Algorithm 1) is O(nm^2).

Algorithm 1. Quad-Tree construction

Tlist: Quad-Tree node list
Tlist = [];
Tlist(1) = BS;
Tind = 1;
while *Tlist* is not empty
    Split the *Tlist*(*Tind*) node into four sub-blocks (*UL*, *UR*, *LL*, and *LR*,) and calculate their attributes.
    If *coverSets* of *Tlist*(*Tind*), *UL*, *UR*, *LL*, and *LR* are all equal
    Tlist(*Tind*).*status* = 1
    Tind = *Tind* + 1
    else
        Remove *Tlist*(*Tind*) node from *Tlist*
        Append *UL*, *UR*, *LL* and *LR* into *Tlist*

In sparser maps, there are more Quad-Tree nodes that do not require subdivision into the most basic grid points. To further reduce the likelihood of Quad-Tree nodes being

subdivided downward, every time the placement position of a pad is determined, the sensors covered by it are excluded from consideration. As the number of sensors to be considered decreases, more Quad-Tree nodes do not require subdivision.

In the methods proposed by Chen et al. [6], the Minimum Set Cover (MSC) algorithm demonstrates good performance in terms of calculation speed and the number of pads used. Therefore, we propose an On-Demand Quad-Tree construction method combined with MSC, as illustrated in Figure 5. The overall procedure involves first executing the On-Demand Quad-Tree construction based on MSC (Algorithm 2) to configure pads at appropriate locations. Then, redundant pads are removed using Removing Redundant Pads (Algorithm 3), which internally calls Connectedness Checking (Algorithm 4) and Coverage Checking (Algorithm 5) to verify whether the connectedness and coverage constraints are satisfied. The details of Algorithms 2–5 are described below.

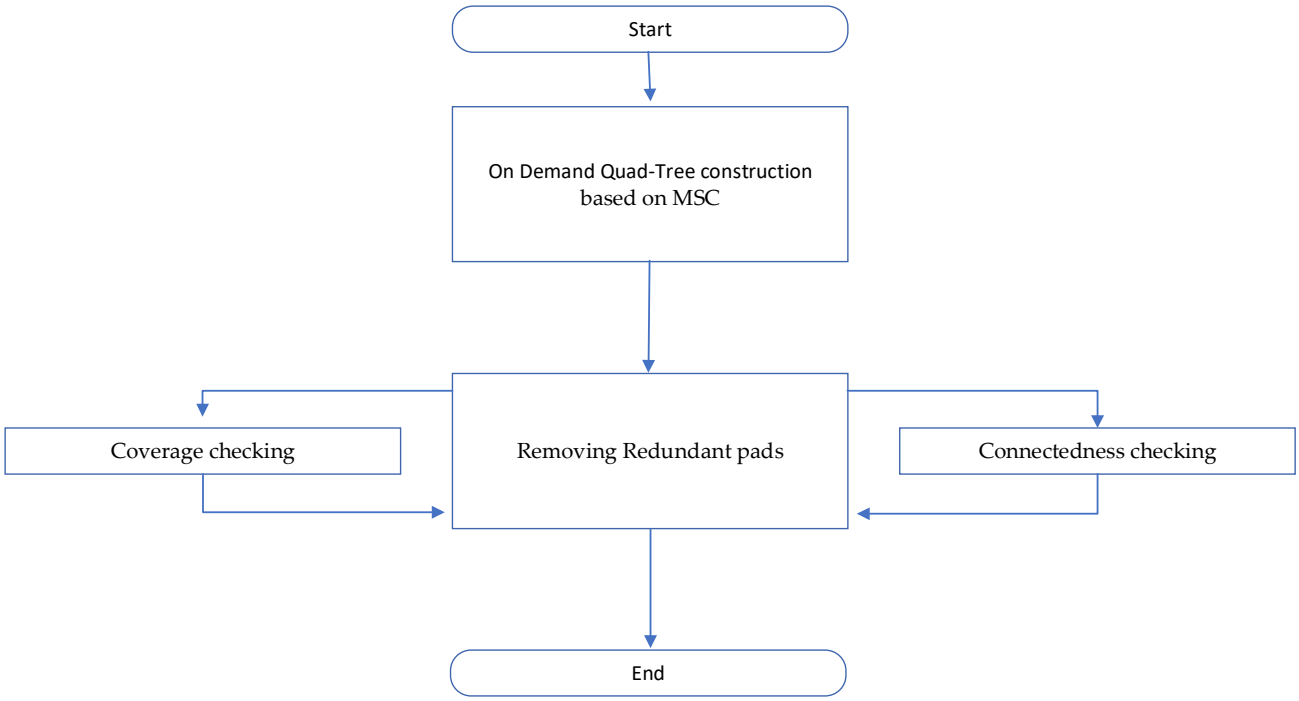


Figure 5. The flowchart of the proposed method.

Algorithm 2. On-Demand Quad-Tree construction based on MSC

Tlist: Quad-Tree node list
Sset: the set all sensors
Tlist = [];
Tlist(1)= BS;
Tind = 1;
pads = [BS];
while *Sset* is not empty
    Find the block *m* with the largest *coverNum* in *Tlist*
    if *Tlist*(*m*).*status* == 0
        Split the *Tlist*(*Tind*) node into four sub-blocks (*UL*, *UR*, *LL*, and *LR*,) and calculate their attributes.
        if *coverSets* of *Tlist*(*m*), *UL*, *UR*, *LL*, and *LR* are all equal
            Tlist(*m*).*status* = 1
        else
            Remove *Tlist*(*m*) node from *Tlist*
            Append *UL*, *UR*, *LL* and *LR* into *Tlist*

Algorithm 2. *Cont.*

else
if the distance between the center position of *Tlist*(*m*) and any pad in *pads* is less than $\boldsymbol{D_{maxpad}}$
Append the center position of *Tlist*(*m*) into pads
Sset = *Sset* − *Tlist*(*m*).*coverSet*
Remove *Tlist*(*m*).*coverSet* from the *coverSet* of all nodes in the *Tlist*
else
if *Tlist*(*m*) is not a basic grid point
Tlist(*m*) is forcibly split into 4 child nodes and added to *Tlist*
Remove *Tlist*(*m*) node from *Tlist*
else
Find the pad *A* which position is closest to *Tlist*(*m*) from pads
Place a pad *B* in the direction of pad A along *Tlist*(*m*). The distance between pad *A* and *B* is $\boldsymbol{D_{maxpad}}$
Append pad *B* into *pads*

The final set of pad placement locations should undergo a final check to identify any redundant pads. Here a redundant pad is one that can be removed without affecting the coverage of all sensors and maintaining connectivity between pads. The algorithms used in this article to perform this check are described below.

Algorithm 3. Removing Redundant pads

Sset: the set all sensors
P: the set all pads
for each pad *p* in *P*
If removing *p* does not affect the connectivity between pads and coverage of sensors, then *P* = *P* − *p*

This algorithm checks whether the removal of each pad affects connectivity and coverage. Assuming $n_s = |S|$, $n_p = |P|$, according to the algorithm analyses below, the computational cost of this algorithm is $\mathrm{O}\left(n_p\left(n_s n_p + n_p^3\right)\right) = \mathrm{O}\left(n_s n_p^4\right)$.

Algorithm 4. Connectedness checking

P: the set all pads
SC = [BS]
while *P* is not empty
Find the pad *p* closest to the element in *SC* from *P*,
if its distance is greater than $\boldsymbol{D_{maxpad}}$
return false
else
SC = *SC* + *p*;
P = *P* − *p*;
return true

Let $n_p = |P|$ represent the number of pads. Since the number of loops in the algorithm is dependent on n_p, and the computational cost of finding the nearest pad in the loop is $\mathrm{O}(n_p^2)$, the overall computational cost of this algorithm is $\mathrm{O}(n_p^3)$.

Algorithm 5. Coverage checking

```
Sset: the set all sensors
P: the set all pads
for each sensor s in Sset
    covered = false
    for each pad p in P
        if the distance between s and p is less or equal to D_maxcharging
            covered = true
            break
    if covered == false
        return false
return true
```

It is evident from the above algorithm that when $n_s = |S|$, $n_p = |P|$, the algorithm computational cost is $O(n_s n_p)$.

4. Simulation Results

To evaluate the merits of the proposed algorithm, we consider a large wireless rechargeable sensor network (WRSN) comprising 50 to 500 sensor nodes deployed evenly and randomly within a rectangular area. The BS is deployed at the center of the rectangular area. The simulations were implemented and executed using MATLAB R2023b, running on a computer with an Intel Core i5-3470 CPU (3.2 GHz) and 16 GB of RAM. It is important to note that all simulation results represent the average of 30 simulations. Other relevant parameters are listed in Table 3.

Table 3. Parameters and values.

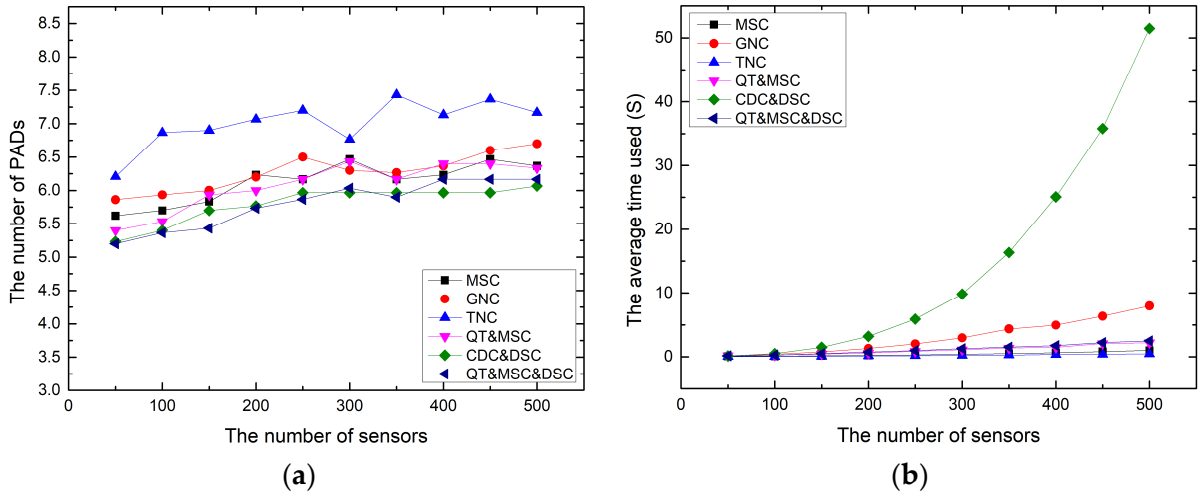
Parameters	Values
E_{sensor}	200 J
E_{max}	1000 J
P_{fly}	10 J/s
V_{fly}	35 m/s

4.1. Performance Comparison

In our simulation, we utilized three types of maps: small-scale (4096 × 4096), medium-sized (6144 × 6144), and large-scale (8192 × 8192). Across these three map scales, we computed the required number of charging pads and the algorithm computation time for six methods: MSC, GNC, TNC, CDC&DSC, our proposed Quad-Tree algorithm (QT&MSC), and QT&MSC&DSC. MSC, GNC, and TNC were proposed by Chen et al. [6]. CDC&DSC was proposed by Chen et al. [13]. Additionally, [6,13] were the only current papers we found that propose a pad allocation algorithm, and the pad configuration obtained using CDC&DSC is quite streamlined. By combining its proposed algorithm for removing duplicate pads (DSC), we propose a version of QT&MSC&DSC and include it in the comparison.

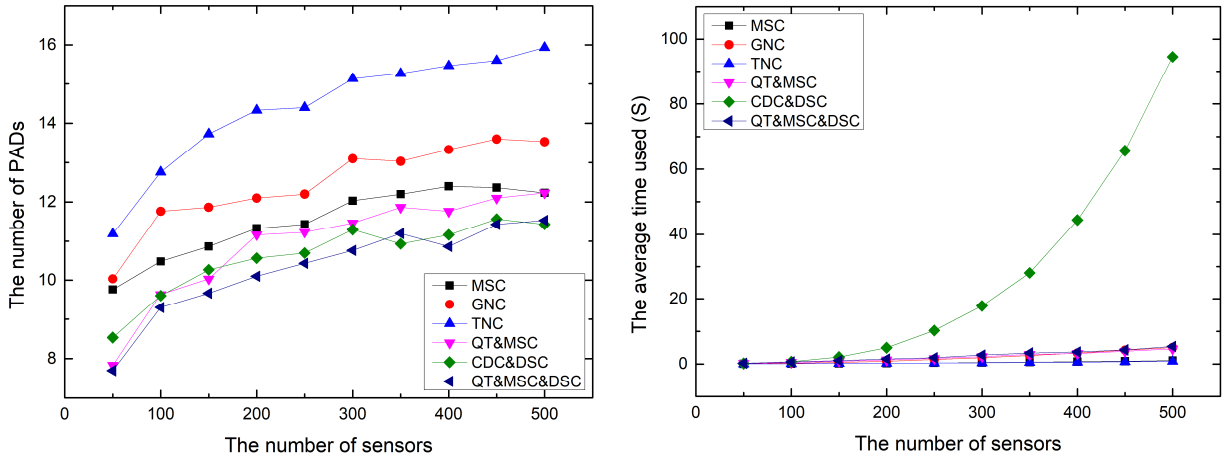
Figures 6–8 depict the performance of different methods on maps of different sizes.

Firstly, in the small map (Figure 6), MSC, GNC, and TNC failed to find an appropriate pad configuration for some maps with only 50 scattered sensors. Conversely, the proposed methods and CDC&DSC encountered no problems with these maps. Without using DSC, the number of pads used by QT&MSC could be reduced by up to 2.92% compared to MSC, with an average reduction of only 0.48%. CDC&MSC exhibited a maximum reduction of 7.73% and an average reduction of 5.11%. When combined with DSC, QT&MSC&DSC achieved reductions of up to 8.02% and an average of 5.05%. Therefore, in terms of reducing the number of pads used, CDC&DSC and QT&MSC&DSC performed similarly, indicating that appropriate removal of redundant pads offers advantages.



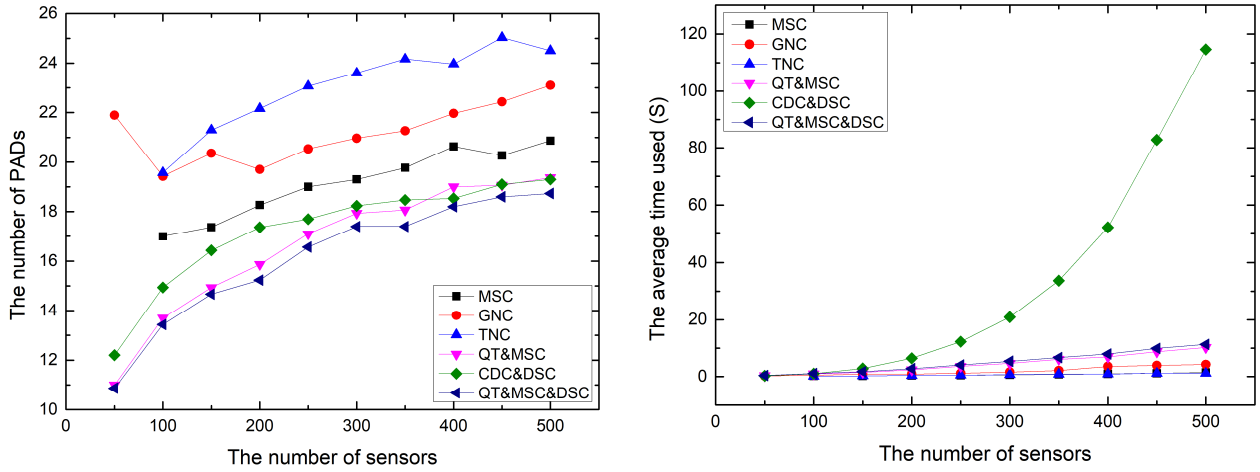
(**a**) (**b**)

Figure 6. Comparisons of (**a**) the required number of pads and (**b**) the average execution time of related methods on small-scale maps.



(**a**) (**b**)

Figure 7. Comparisons of (**a**) the required number of pads and (**b**) the average execution time of related methods on medium-scale maps.



(**a**) (**b**)

Figure 8. Comparisons of (**a**) the required number of pads and (**b**) the average execution time of related methods on large-scale maps.

However, Figure 6b illustrates the high computational complexity associated with DSC. CDC requires appropriate determination of the number of clusters. In our simulation environment, CDC generated a significant number of clusters, leading to prolonged processing time for DSC. Compared to MSC, the average computation time increased by 2588.89%. The issue with QT&MSC is relatively minor. Compared to MSC, QT&MSC experienced an average increase in computation time of 164.44%, while QT&MSC&DSC, which incorporates DSC, experienced an average increase of 201.43%.

As the network size expands to 6144 × 6144, the gap between the proposed methods and MSC widens slightly. During the simulations, it was observed that MSC, TNC, and GNC algorithms fail when nodes are distributed far apart, as they struggle to find suitable sensor locations meeting the adjacency conditions from the base station. For 50 sensors, MSC, GNC, and TNC encountered problems in finding appropriate pad configurations in some maps. Additionally, MSC and TNC also faced issues when there were 100 sensors. However, because GNC utilizes flight range to choose the next optimal position, resulting in larger coverage compared to using the two-hop method as in MSC and TNC, GNC did not encounter similar problems with 100 sensors.

There were no issues with the proposed methods and CDC&DSC, as these algorithms consider that when the relevant adjacent position cannot be found, pads are filled in at the appropriate positions and connected to the nearest candidate pad position. Excluding the cases of 50 or 100 sensors, without using DSC, the number of pads used by QT&MSC could be reduced by up to 7.67% compared to MSC, with an average reduction of 3.74%. CDC&MSC exhibited a maximum reduction of 7.73% and an average of 7.40%. QT&MSC&DSC decreased by 12.37% at most and 9.59% on average. Therefore, in terms of reducing the number of pads used, QT&MSC is slightly less effective than CDC&DSC, while QT&MSC&DSC performs better than CDC&DSC.

However, Figure 7b again highlights the problem of high computational complexity of DSC. Compared with MSC, CDC&DSC increases the calculation time by an average of 4137.68%. QT&MSC and QT&MSC&DSC, respectively, increase the calculation time by an average of 383.26% and 463.92% compared to MSC.

When the network scale expands to 8192 × 8192, the gap between the proposed methods and MSC widens. As analyzed previously, in environments with 50 to 150 sensors, MSC, GNC, and TNC encounter problems with some maps being unable to find suitable pad configurations. Conversely, our proposed methods and CDC&DSC have no such issues. Excluding scenarios with 50, 100, and 150 sensors, CDC&DSC can reduce the number of pads by up to 10.18%, with an average reduction of 7.21% compared to MSC. Without using DSC, the number of pads used by QT&MSC can be reduced by up to 13.14%, and on average by 10.37%, compared to MSC. The performance of QT&MSC exceeds that of CDC&MSC. Moreover, after combination with DSC, QT&MSC&DSC achieves a maximum reduction of 16.61% and an average reduction of 13.11%. In large-scale network environments, both QT&MSC and QT&MSC&DSC surpass CDC&DSC. The reduction ratio of the average number of pads compared to MSC in CDC&DSC is slightly lower than that in the medium-sized network environment. The average reduction ratio of QT&MSC increases significantly, indicating that QT&MSC is more suitable for large-scale network environments than CDC&DSC.

Figure 8b illustrates that CDC&DSC increases the calculation time by 3727.14% on average compared to MSC. QT&MSC and QT&MSC&DSC, respectively, increase the calculation time by 591.29% and 687.13% on average compared to MSC. This is because QT requires more network cutting time for larger network areas and considers more possible locations for placing pads, resulting in better results than MSC, which only considers the locations of sensors. CDC places pads at the center of each cluster after clustering, so the result of clustering determines the quality of the resulting pad configuration.

Figure 8 illustrates that in large-size maps, the performance gap between QT&MSC and CDC&DSC tends to widen. Therefore, an additional performance comparison of QT&MSC, CDC&DSC, and QT&MSC&DSC under extremely large maps (16,384 × 16,384)

is included. Since MSC, GNC, and TNC cannot complete pad configuration in this environment, these three methods are excluded from this part of the experiment. In the 8192 × 8192 environment, QT&MSC achieves a maximum reduction of 9.84% and an average reduction of 3.39% compared to CDC&DSC, while QT&MSC&DSC achieves a maximum reduction of 10.93% and an average reduction of 6.35%. Figure 9 shows that in a 16,384 × 16,384 network environment, the number of pads used by QT&MSC is reduced by up to 9.97% and an average reduction of 9.24% compared to CDC&DSC, while QT&MSC&DSC is reduced by up to 12.05% and an average reduction of 10.82%. On average, the proposed methods tend to perform better than CDC&DSC. Additionally, in terms of usage time, compared to CDC&DSC, the average reduction time of QT&MSC is reduced from 68.70% in Figure 8 to 45.02% in Figure 9, and the average reduction time of QT&MSC&DSC is reduced from 64.12% to 38.18%. This indicates that as the graph size increases, the need for a Quad-Tree to crop the graph also increases, reducing the time advantage of CDC&DSC.

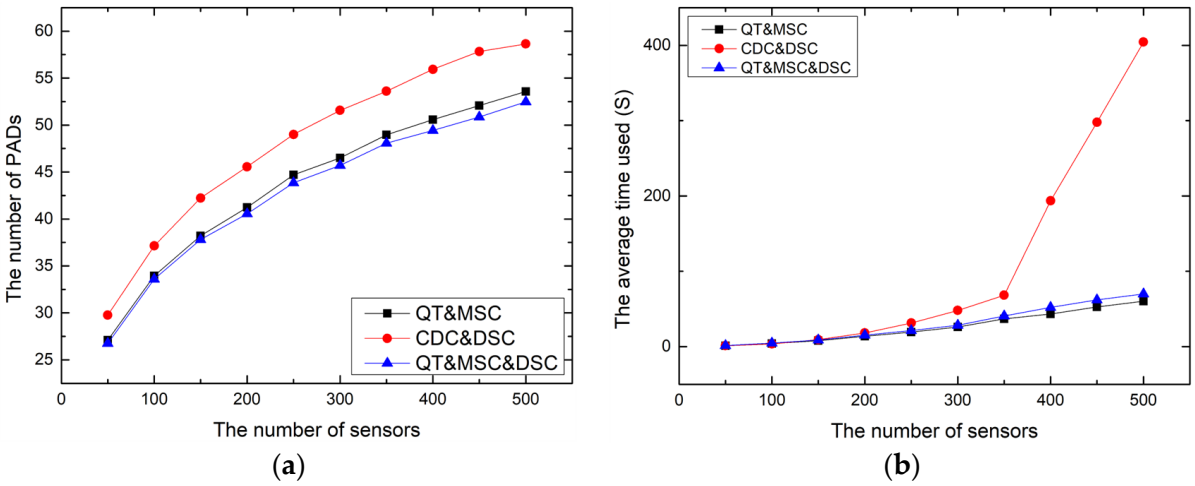


Figure 9. Comparisons of (**a**) the required number of pads and (**b**) the average execution time of QT&MSC, QT&MSC&DSC, CDC&MSC methods on extremely large-scale maps.

In the above experiments, it can be observed that as the sensor density increases, there is a trend of increased pad usage for each method. Additionally, the rate of increase in pad usage tends to slow as sensor density rises. Theorem 2 in reference [6] proposes that for an l by m square area, the upper limit of pad usage is given by $\left\lceil \frac{2l}{\sqrt{2}d_{maxpad}} \right\rceil \left\lceil \frac{2m}{\sqrt{2}d_{maxpad}} \right\rceil$. Furthermore, since the base station is located at the center of the map in this study, the upper limit of pad usage should be revised to $(2\left\lceil \frac{l/2}{d_{maxpad}/\sqrt{2}} \right\rceil)(2\left\lceil \frac{m/2}{d_{maxpad}/\sqrt{2}} \right\rceil)$. For the small, medium, large, and xlarge maps mentioned above, the upper limits of pad usage are 16, 16, 36, and 100, respectively. The number of pads used by each method for various sensor counts is also below these upper limits, which can indeed serve as a reference upper limit calculation formula. For the proposed method, QT&MSC, when the number of sensors in the network is increased to 5000, the number of pads used for the small, medium, large, and extremely large maps is only 46.25%, 81.88%, 71.67%, and 77.30% of the aforementioned upper limits, respectively. Therefore, finding a more precise upper limit will be a direction for future research.

4.2. Performance Comparison of Special Test Maps

To observe the differences between the results obtained by the proposed method and the optimal ones, we generated several sets of test maps with known optimal solutions using the architecture shown in Figure 10. Initially, we positioned sensors (1–4) in the corners. Due to flying distance constraints, placing sensor 1 requires the use of at least pads 1 and 2 to connect it with the base station. Nevertheless, placing any sensors within the

$D_{maxcharging}$ range of the base station, pad 1, or pad 2 does not require additional pads (the same applies to other corner sensors).

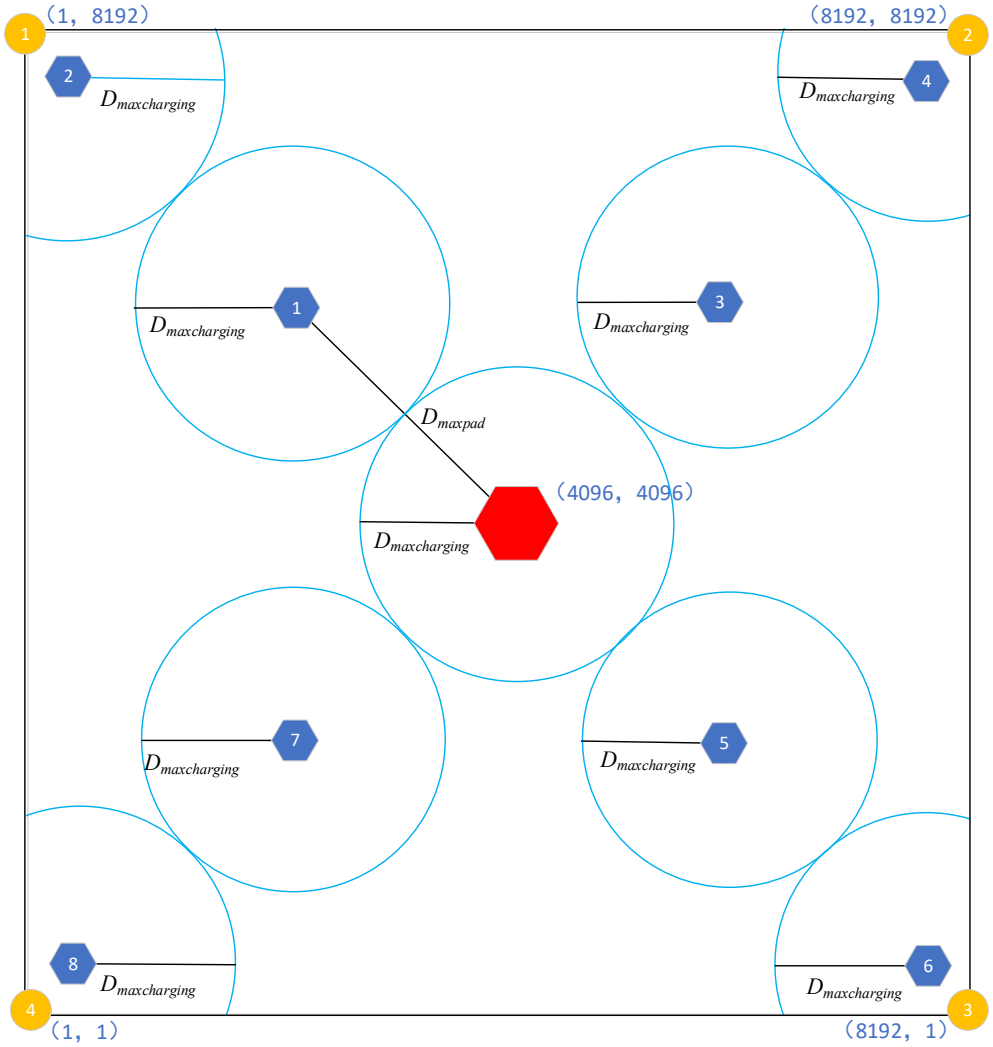


Figure 10. The basic structure for generating the special test maps.

In this section, all generated test maps are square network areas of size 8192 × 8192, with the base station located at the center. The test maps are divided into four main groups. The first to fourth groups of test maps, respectively, include sensors placed in one to four corners, with the remaining sensors randomly scattered within a specified range according to the number of sensors required. Each group of maps is further subdivided into subcategories ranging from 50, 100, 150, 200, ..., to 500 sensors, with 30 maps per subcategory. For example, in the case of the second group with 50 sensors, sensor 1 and sensor 2 are fixed at positions (1, 8192) and (8192, 8192), respectively. The remaining 48 sensors are then randomly distributed within circular areas centered around pad 1, pad 2, pad 3, pad 4, or the base station, with a radius of $D_{maxcharging}$. Of course, all sensors must be located within the 8192 × 8192 network area. Therefore, the total number of test maps in this section is 4 × 10 × 30. Additionally, the optimal number of pads used in the first to fourth groups of test maps is 3, 5, 7, and 9, respectively.

While the optimal number of pads remains constant within the same set of maps regardless of the number of sensors, the results displayed in Figures 11–14 indicate that, in general, the number of pads used increases as the number of sensors in the network increases. This could be attributed to the fact that with more sensors, there are fewer positions that can cover all sensors simultaneously, leading the algorithms to require more

pads to cover all sensors due to deviations in pad placement from the optimal positions. As the complexity of sensor distribution increases from the first group of maps to the fourth group, there is a trend of increasing ratios in the number of pads used by the MSC, GNC, and TNC methods compared to the optimal solution. For maps with 500 sensors, the ratio of pads used by MSC compared to the optimal solution increases from 133.33% to 142.59%, for GNC from 137.78% to 157.04%, and for TNC from 193.33% to 215.19%. In contrast, CDC&DSC remains relatively stable, increasing from 152.22% to 154.81%. However, the method proposed in this paper, QT&MSC, decreases from 124.44% to 114.07%. In terms of computation time, QT&MSC only increases the time by 4.10% to 60.65% compared to MSC, but it is only 0.88% to 1.59% of the time taken by CDC&MSC. The number of pads obtained by QT&MSC is closer to the optimal solution compared to other methods, which may be attributed to its global optimization effect achieved by the QT&MSC method, which subdivides the network as needed from a global perspective.

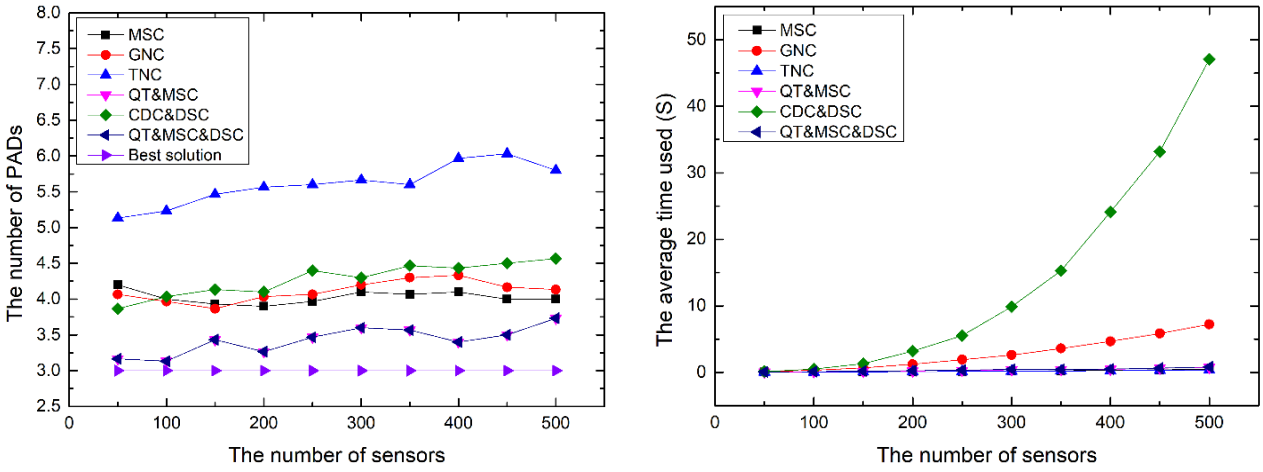


Figure 11. Comparisons of (**a**) the required number of pads and (**b**) the average execution time of related methods on the first group of special test maps.

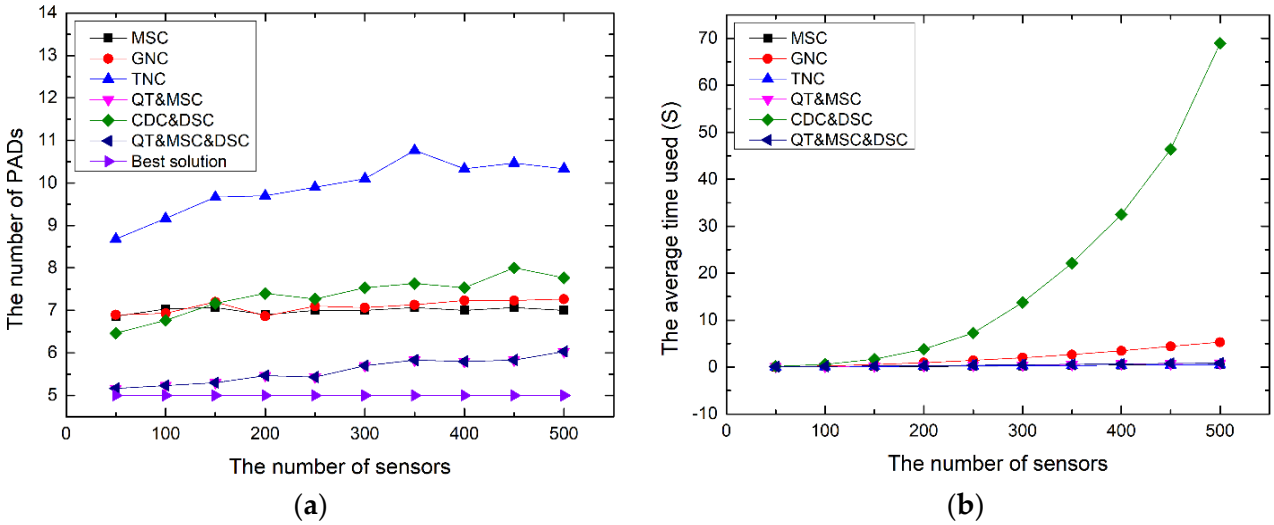


Figure 12. Comparisons of (**a**) the required number of pads and (**b**) the average execution time of related methods on the second group of special test maps.

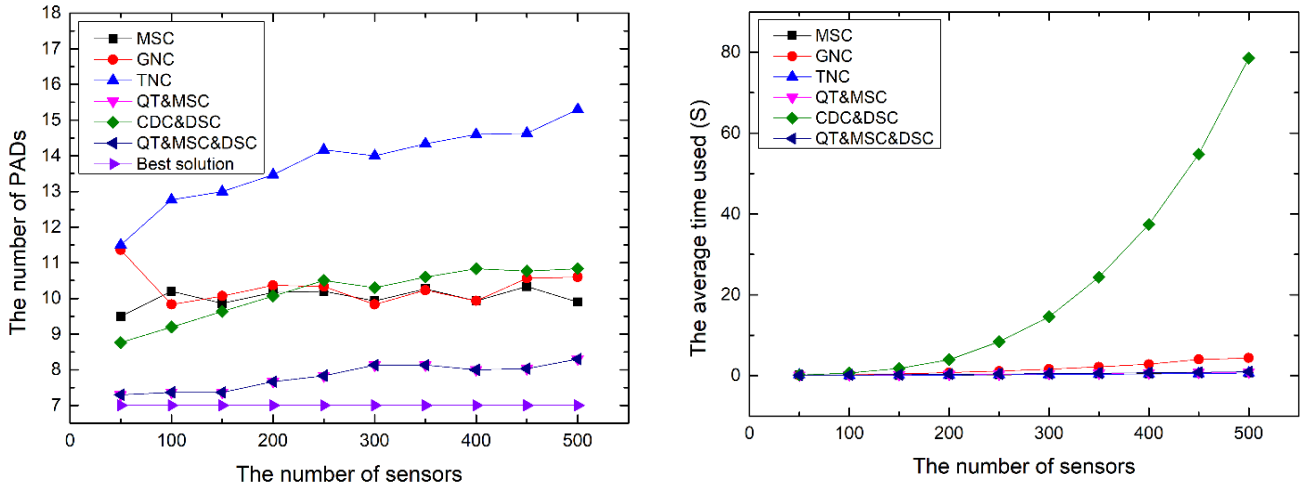


Figure 13. Comparisons of (**a**) the required number of pads and (**b**) the average execution time of related methods on the third group of special test maps.

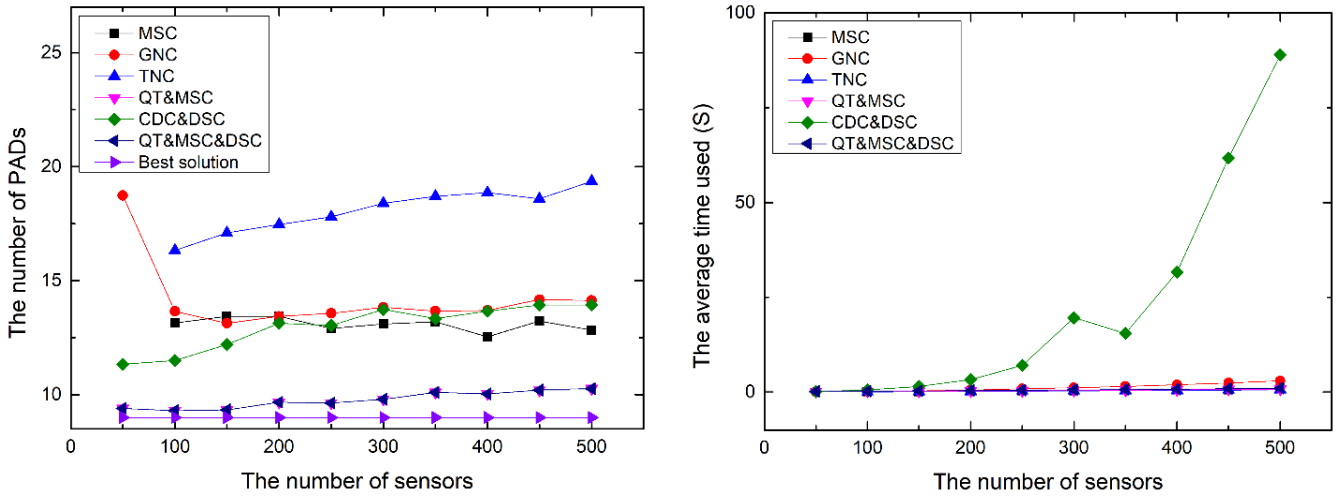


Figure 14. Comparisons of (**a**) the required number of pads and (**b**) the average execution time of related methods on the fourth group of special test maps.

Observing the above results, it is apparent that there is a trend of increased pad usage for each method as the number of sensors increases across different levels of sensor distribution complexity (from group 1 to group 4). However, this trend is evidently less significant compared to the results in Section 4.1. Therefore, it can be preliminarily inferred that the difficulty each method faces in finding optimal solutions, as mentioned in the second possible reason proposed in Section 4.1, should be relatively minor.

4.3. Impact on Changing the Size of the Minimum Unit of the Quad-Tree

According to the previous simulation results, it is found that the Quad-Tree takes more time to partition large-size maps. During the Quad-Tree partitioning process, the minimum unit will not be further divided. Therefore, increasing the size of the minimum unit of Quad-Tree and ending Quad-Tree division earlier should reduce the time required for Quad-Tree division and also reduce the number of positions that need to be considered during subsequent MSC runs, thus further reducing the algorithm running time. However, due to insufficient partitioning, the optimal pad placement may have been overlooked,

increasing the number of pads used in the resulting pad configuration. The following simulation uses QT&MSC-n to represent the performance of the QT&MSC algorithm when the minimum unit is set to n meters by n meters in size. The original QT&MSC is used as the comparison baseline, and its minimum division unit is 1 m by 1 m.

Figure 15 illustrates the performance of QT&MSC with varying minimum Quad-Tree units in a 4096 × 4096 network environment. The data indicate that as the minimum Quad-Tree unit size increases, the number of pads required for QT&MSC also increases, ranging from 1.63% to 27.25%. Simultaneously, the reduction in execution time also rises from 7.38% to 61.00%, aligning with the anticipated impact of larger Quad-Tree units.

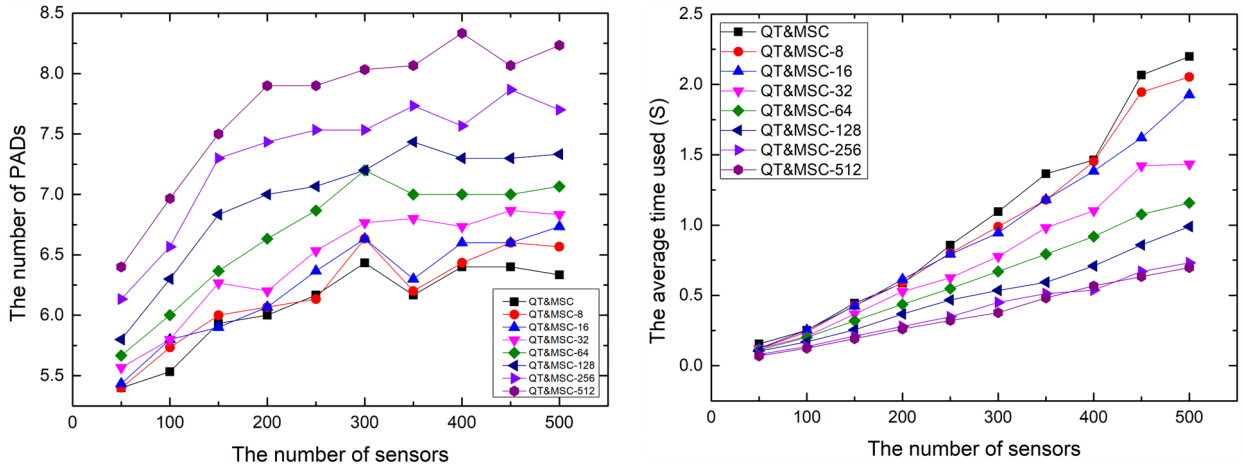


Figure 15. Comparisons of (**a**) the required number of pads and (**b**) the average execution time of the proposed QT&MSC method while changing the minimal unit size of QT on small-scale maps.

In a network environment of 6144 × 6144, Figure 16 illustrates that as the minimum Quad-Tree unit increases, the number of pads required for QT&MSC rises from 0.91% to 21.42%. Simultaneously, the reduction in execution time increases from 4.23% to 53.58%. Similar trends are observed in the 8192 × 8192 maps depicted in Figure 17, where the number of pads required for QT&MSC increases from 1.09% to 23.70% with the increasing minimum Quad-Tree unit, while the reduction in execution time grows from 4.41% to 62.07%.

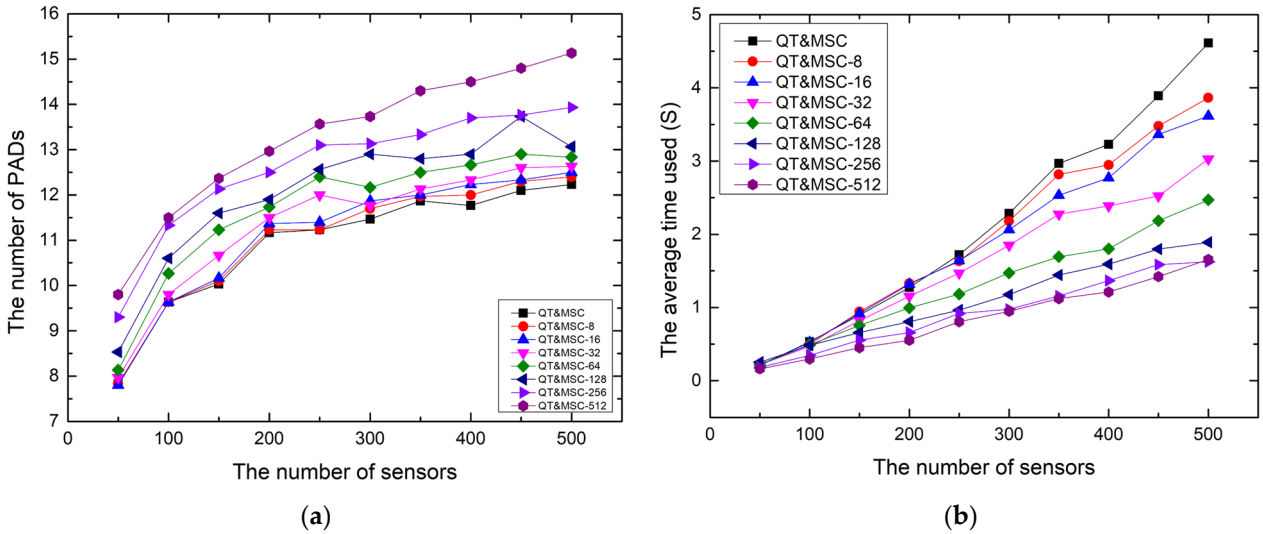
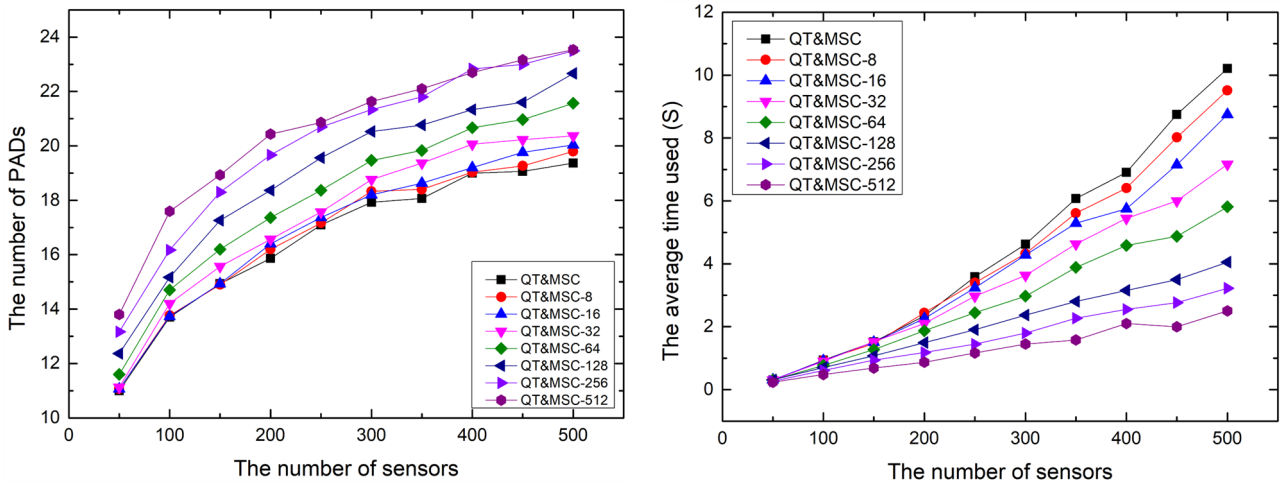


Figure 16. Comparisons of (**a**) the required number of pads and (**b**) the average execution time of the proposed QT&MSC method while changing the minimal unit size of QT on medium-scale maps.



(**a**) (**b**)

Figure 17. Comparisons of (**a**) the required number of pads and (**b**) the average execution time of the proposed QT&MSC method while changing the minimal unit size of QT on large-scale maps.

Overall, across the three different map sizes, QT&MSC-8 consistently reduces computation time by approximately 4% to 7% while increasing the pad count by around 1%. On the other hand, QT&MSC-512 saves over 50% of execution time but increases the pad count by more than 20%. Further exploration and analysis on how to select the optimal size of Quad-Tree minimum units will be one of our future research directions.

5. Conclusions

In this work, we introduced a novel approach utilizing the Quad-Tree concept to address the wireless charging pad deployment problem, aiming to reduce the number of deployed pads. However, our proposed scheme may overlook the optimal pad placement due to insufficient segmentation, potentially leading to an increase in the number of pads required in the resulting configuration. Conversely, reducing the minimum unit of the Quad-Tree could increase the time needed for Quad-Tree division and also expand the number of positions that need to be considered, thus potentially prolonging the algorithm's running time. Therefore, further exploration and theoretical analysis are needed to determine the optimal size of Quad-Tree segmentation units. Additionally, determining a more precise upper bound on the number of pads used by the proposed method, considering the characteristics of the Quad-Tree, will be a direction for our future research.

Furthermore, we recognize that other Quad-Tree-like geometric methods may offer additional opportunities for devising new pad deployment schemes in the future. These alternative methods could potentially enhance the efficiency and effectiveness of wireless charging pad deployment, warranting further investigation and research.

The values of $D_{maxcharging}$ and D_{maxpad} of every specific drone are quite ideal and different due to different battery size, flying speed, power consumption rates, charging speed, charging position and direction, etc. However, in real conditions, the two adopted system parameters should be much less than $D_{maxcharging}$ and D_{maxpad} to guarantee the coverage and connectivity property of the constructed flight network. After adopting such large system parameters, the number of required pads and the costs would increase by applying our proposed algorithm. Another issue is the delay time between when a sensor issues a charging request and a drone visits (from the base station) and starts to charge the desired sensor, which should be shorter than the waiting time the sensor can accept. Similarly, when considering the delay time issue, either a sensor should issue its charging request early, or more pads need to be deployed to shorten the maximum shortest path between the sensor and base station in the network to meet the needs of the actual situations.

Author Contributions: Conceptualization, R.-H.C. and Z.-L.Z.; methodology, C.-W.Y.; software, R.-H.C.; validation, Z.-L.Z. and C.-W.Y.; writing—original draft preparation, Z.-L.Z. and R.-H.C.; writing—review and editing, Z.-L.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by Wenzhou Science and Technology Commissioner Project (Grant No. X2023012), in part by 2023 Zhejiang Province Industry University Cooperation Collaborative Education Project (Grant No. 323), in part by the first batch of teaching innovation teams for teachers in higher vocational colleges in Wenzhou (Grant No. 1).

Data Availability Statement: Data are contained within the article.

Acknowledgments: The authors would like to thank Chung Hua University for supporting this research.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Almomani, A.; Mahmoud, M.A.; Ahmad, M.S. Factors That Influence the Acceptance of Internet of Things Services by Customers of Telecommunication Companies in Jordan. *J. Organ. End User Comput.* **2018**, *30*, 51–63. [CrossRef]
2. Xie, L.; Shi, Y.; Hou, Y.T.; Lou, W.; Sherali, H.D.; Midkiff, S.F. Multi-Node Wireless Energy Charging in Sensor Networks. *IEEE/ACM Trans. Netw.* **2015**, *23*, 437–450. [CrossRef]
3. Fu, L.; He, L.; Cheng, P.; Gu, Y.; Pan, J.; Chen, J. ESync: Energy Synchronized Mobile Charging in Rechargeable Wireless Sensor Networks. *IEEE Trans. Veh. Technol.* **2015**, *65*, 7415–7431. [CrossRef]
4. Lin, C.; Zhou, J.; Guo, C.; Song, H.; Wu, G.; Obaidat, M.S. TSCA: A Temporal-Spatial Real-Time Charging Scheduling Algorithm for On-Demand Architecture in Wireless Rechargeable Sensor Networks. *IEEE Trans. Mob. Comput.* **2017**, *17*, 211–224. [CrossRef]
5. Long, N.T.; Huong, T.T.; Bao, N.N.; Binh, H.T.T.; Le Nguyen, P.; Nguyen, K. Q-learning-based distributed multi-charging algorithm for large-scale WRSNs. *Nonlinear Theory Its Appl. IEICE* **2023**, *14*, 18–34. [CrossRef]
6. Chen, J.; Yu, C.W.; Ouyang, W. Efficient Wireless Charging Pad Deployment in Wireless Rechargeable Sensor Networks. *IEEE Access* **2020**, *8*, 39056–39077. [CrossRef]
7. Qureshi, B.; Aziz, S.A.; Wang, X.; Hawbani, A.; Alsamhi, S.H.; Qureshi, T.; Naji, A. A state-of-the-art Survey on Wireless Rechargeable Sensor Networks: Perspectives and Challenges. *Wirel. Netw.* **2022**, *28*, 3019–3043. [CrossRef]
8. Su, C.; Ye, F.; Wang, L.-C.; Wang, L.; Tian, Y.; Han, Z. UAV-Assisted Wireless Charging for Energy-Constrained IoT Devices Using Dynamic Matching. *IEEE Internet Things J.* **2020**, *7*, 4789–4800. [CrossRef]
9. Li, S.; Wang, A.; Sun, G.; Liu, L. Improving charging performance for wireless rechargeable sensor networks based on charging UAVs: A joint optimization approach. In Proceedings of the 2020 IEEE Symposium on Computers and Communications (ISCC), Rennes, France, 7–10 July 2020; pp. 1–7. [CrossRef]
10. Yang, L.; Su, Z.; Yang, H.; Na, Z.; Yan, F. An Efficient Charging Algorithm for UAV-aided Wireless Sensor Networks. In Proceedings of the 2020 IEEE 6th International Conference on Computer and Communications (ICCC), Chengdu, China, 11–14 December 2020; pp. 834–838. [CrossRef]
11. Yoon, I. Data Acquisition Control for UAV-Enabled Wireless Rechargeable Sensor Networks. *Sensors* **2023**, *23*, 3582. [CrossRef] [PubMed]
12. Rahman, S.; Akter, S.; Yoon, S. Energy-efficient charging of sensors for UAV-aided wireless sensor network. *Int. J. Internet Broadcast. Commun.* **2022**, *14*, 80–87. [CrossRef]
13. Chen, Y.; Gu, Y.; Li, P.; Lin, F. Minimizing the Number of Wireless Charging PAD for UAV-Based Wireless Rechargeable Sensor Networks. *Int. J. Distrib. Sens. Netw.* **2021**, *17*, 15501477211055958. [CrossRef]
14. Frisken, S.F.; Perry, R.N. Simple and Efficient Traversal Methods for Quadtrees and Octrees. *J. Graph. Tools* **2002**, *7*, 1–11. [CrossRef]
15. Choi, G.-H.; Lee, W.; Kim, T.-W. Voyage optimization using dynamic programming with initial quadtree based route. *J. Comput. Des. Eng.* **2023**, *10*, 1185–1203. [CrossRef]
16. Lu, X.; Wang, P.; Niyato, D.; Kim, D.I.; Han, Z. Wireless networks with RF energy harvesting: A contemporary survey. *IEEE Commun. Surv. Tutor.* **2014**, *17*, 757–789. [CrossRef]
17. Xie, L.; Shi, Y.; Hou, Y.T.; Lou, A. Wireless power transfer and applications to sensor networks. *IEEE Wirel. Commun.* **2013**, *20*, 140–145. [CrossRef]
18. Joo, C.; Kim, T. The Efficiency Improvement of Track-Type Wireless Power Transmission Systems through Electromagnetic Finite Element Analysis. *Energies* **2023**, *16*, 8045. [CrossRef]
19. Chen, J.; Chen, H.; Ouyang, W.; Yu, C.W. A Temporal and Spatial Priority With Global Cost Recharging Scheduling in Wireless Rechargeable Sensor Networks. *Int. J. Grid High Perform. Comput.* **2022**, *14*, 1–31. [CrossRef]
20. Nguyen, P.L.; La, V.Q.; Nguyen, A.D.; Nguyen, T.H.; Nguyen, K. An on-demand charging for connected target coverage in WRSNs using fuzzy logic and Q-learning. *Sensors* **2021**, *21*, 5520. [CrossRef]
21. Chen, T.-S.; Chen, J.-J.; Gao, X.-Y.; Chen, T.-C. Mobile Charging Strategy for Wireless Rechargeable Sensor Networks. *Sensors* **2022**, *22*, 359. [CrossRef]

22. Zhong, P.; Xu, A.; Zhang, S.; Zhang, Y.; Chen, Y. EMPC: Energy-minimization path construction for data collection and wireless charging in WRSN. *Pervasive Mob. Comput.* **2021**, *73*, 101401. [CrossRef]
23. Li, Y.; Zhong, L.; Lin, F. Predicting-scheduling-Tracking: Charging nodes with non-deterministic mobility. *IEEE Access* **2021**, *9*, 2213–2228. [CrossRef]
24. Zhu, J.; Feng, Y.; Liu, M.; Chen, G.; Huang, Y. Adaptive Online Mobile Charging for Node failure Avoidance in Wireless Rechargeable Sensor Networks. *Comput. Commun.* **2018**, *126*, 28–37. [CrossRef]
25. Jin, Y.; Xu, J.; Wu, S.; Xu, L.; Yang, D.; Xia, K. Bus network assisted drone scheduling for sustainable charging of wireless rechargeable sensor network. *J. Syst. Archit.* **2021**, *116*, 102059. [CrossRef]
26. Wu, P.; Xiao, F.; Sha, C.; Huang, H.; Sun, L. Trajectory Optimization for UAVs' Efficient Charging in Wireless Rechargeable Sensor Networks. *IEEE Trans. Veh. Technol.* **2020**, *69*, 4207–4220. [CrossRef]
27. Liang, S.; Fang, Z.; Sun, G.; Lin, C.; Li, J.; Li, S.; Wang, A. Charging UAV deployment for improving charging performance of wireless rechargeable sensor networks via joint optimization approach. *Comput. Netw.* **2021**, *201*, 108573. [CrossRef]
28. Lin, C.; Yang, W.; Dai, H.; Li, T.; Wang, Y.; Wang, L.; Wu, G.; Zhang, Q. Near Optimal Charging Schedule for 3-D Wireless Rechargeable Sensor Networks. *IEEE Trans. Mob. Comput.* **2021**, *22*, 3525–3540. [CrossRef]