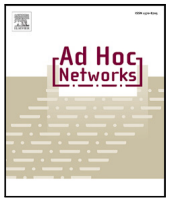
Ad Hoc Networks 138 (2023) 103017

Contents lists available at ScienceDirect

Ad Hoc Networks

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



Collecting large volume data from wireless sensor network by drone

Rone Ilídio da Silva [a,*], Josiane Da Costa Vieira Rezende [b], Marcone Jamilson Freitas Souza [c]

[a] Department of Technology in Civil Engineering, Computing, Automation, Telematics and Humanities (DTECH) - Federal University of Sao Joao del-Rei (UFSJ), Rodovia MG 443, s/n, km 7, Ouro Branco, 36.490-972, Minas Gerais, Brazil

[b] Fumec University – Minas Gerais, Rua Cobre, 200, Bairro Cruzeiro, Belo Horizonte, 30.310-190, Minas Gerais, Brazil

[c] Department of Computer Science (DECOM) - Federal University of Ouro Preto (UFOP), Campus Universitario Morro do Cruzeiro, Ouro Preto, 35,400-000, Minas Gerais, Brazil

ARTICLE INFO

Keywords:
Wireless sensor network
Mobile sink
Path planning
UAV
Drone

ABSTRACT

Data collection is the most important task in wireless sensor networks (WSN). Each sensor node has to send the sensed data to a special node called sink, which is the user interface. The sensor nodes far from the sink send data to intermediate nodes that forward it by multi-hop data paths. This characteristic leads to higher energy consumption in the sensor nodes close to the sink since they have to relay data from all other sensor nodes. The literature presents several studies that use mobile sinks for data collection to reduce the number of hops in the data paths and distributes the energy consumption, considering that the nodes close to the mobile sink change. However, the majority of these studies consider only the network limitation, such as energy. Furthermore, they also consider sensor nodes sending only one data packet to the mobile sink. This work assumes a quad-copter drone as a mobile sink and sensor nodes having several data packets to send to the sink. We propose two GRASP-based heuristics to define drone tours for data collection. Since this vehicle has limited flight time, the primary metric analyzed here is the overall data collection time. Furthermore, they guarantee that the mobile sink will stay a minimal time inside the radio range of each sensor node to ensure that all of them will have enough time to send all data. The heuristics achieve this guarantee by looking for a subset of locations, among the infinite points inside the monitored area, where the drone will hover for data gathering. Hence, the proposed heuristics have to search for good locations to reduce the data gathering time and define the shortest path to reduce the trip time. Simulated experiments showed that the proposed GRASP-based heuristics outperformed the greed algorithm found as state of the art for this type of scenario, mainly when the volume of data stored in each sensor node is high.

1. Introduction

Wireless Sensor Network (WSN) is a computer network conventionally used to sense environmental data in regions where the human presence is dangerous or impossible. It is composed of tiny devices named sensors nodes. These devices have a set of sensors to sense data, such as temperature, humidity, light level, and substances concentration. Sensor nodes have limited processing power and memory, radios for short-range communication, and batteries as energy supply. The main issue for every application of WSN is energy consumption. Since these devices are placed in dangerous regions, replacing batteries is a very hard or impossible operation. Hence, all applications have to save energy to extend the network lifetime [1].

WSN has a special node called *base station* or *sink* that is the user interface. All nodes send sensed data to the sink. Nodes out of the radio range of the sink create multi-hop routes to forward data, i.e., the intermediate nodes relay data packets toward the sink [2]. However, nodes close to the sink have more energy consumption than the others since they have to relay packets from all the nodes in the network. This is called *hotspot problem* and can lead to a complete network disconnection [3]. Fig. 1 exemplifies the topology, the multi-hop data routes, and the hotspot problem in WSN.

Literature presents several studies that mitigate the hotspot problem by using mobile sinks, such as Chang et al. [4], Sapre and Mini [5], Srivastava et al. [6], Mehto et al. [7], Preetha et al. [8] and Ghorpade and Vijaykarthik [9]. These mobile nodes can move inside the monitored area to collect data from the other sensor nodes. This strategy reduces energy consumption since it decreases the number of hops in data paths. Furthermore, it also distributes the energy consumption among the nodes, given that the closest nodes to the sink change with the sink movement. However, the majority of these studies consider only the WSN limitations. The mobile collector considered by them has no limitations, such as trip distance or time. Moreover, they also consider each sensor node having only a small packet to send to the

* Corresponding author.
E-mail address: rone@ufsj.edu.br (R.I. da Silva).

https://doi.org/10.1016/j.adhoc.2022.103017
Received 15 December 2021; Received in revised form 12 September 2022; Accepted 4 October 2022
Available online 13 October 2022

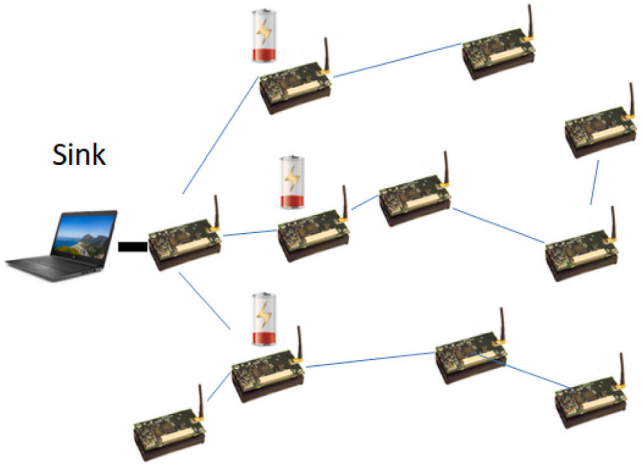


Fig. 1. WSN topology — sensor nodes creating multi-hop routes to forward data to sink node and nodes close to the sink consuming more energy than others.

mobile sink. Hence, the mobile sink needs just to pass over the region covered by the radio of each sensor node.

Studies presented by Saxena et al. [10] and Raj et al. [11] consider unmanned aerial vehicles (UAV) as the mobile sink. They proposed algorithms to reduce UAV trajectories for data gathering. In their scenarios, the wsn defines the cluster-heads, and each trajectory is a sequence of cluster-head locations. Hence, these algorithms solve the well-known Traveling Salesman Problem (TSP) [12]. [13,14] consider a quad-copter drone as the mobile sink in WSN and nodes storing large volume data. Consequently, the drone needs a minimum time inside the radio range of each sensor node to receive all data packets. This type of unmanned vehicle can fly and hover over all monitored areas. They have batteries as the power supply. Therefore, it has limited flight time. Hence, the network and drone limitations must be considered to increase the WSN performance. These works focused on reducing the time spent by the drone to collect data from all sensor nodes. They proposed two algorithms to create the sequence of locations (called tour or trajectory) that the drone has to follow and hover over for data collection. The wsn does not create clusters. Hence, these algorithms must first choose a subset of rendezvous points among the infinite number of points inside the monitored area and then create the trajectory by solving the TSP. They also define a subset of nodes that send data to the drone over each hovering location and another subset of nodes to send data during the drone trip between each two consecutive hovering locations in the tour. However, these algorithms have greedy strategies to define tours.

This work proposes two heuristic algorithms based on the Grasp (Greedy Randomized Adaptive Search Procedures) metaheuristic, defined by Resende [15], to create drone tours for data collection in WSN, which are our main contribution. They focus on reducing the drone flight time for data collection. We consider a quad-copter drone as the mobile sink and sensor nodes having large volume data to be collected. Experiments showed that the proposed heuristics overperformed all the before mentioned greed algorithms in practically all scenarios. This work also presents experiments to determine the best parameters to configure the proposed heuristics to achieve better performance.

The remainder of this text is organized as follows. The next section defines the research problem and presents a table with the notation used here. Section 3 presents a collection of studies related to mobile sink in WSN. We focused on studies that define tours or trajectories to mobile sinks. Section 4 describes the proposed Grasp-based algorithms. Section 5 reports the experiments. Finally, Section 6 brings the conclusions and future works.

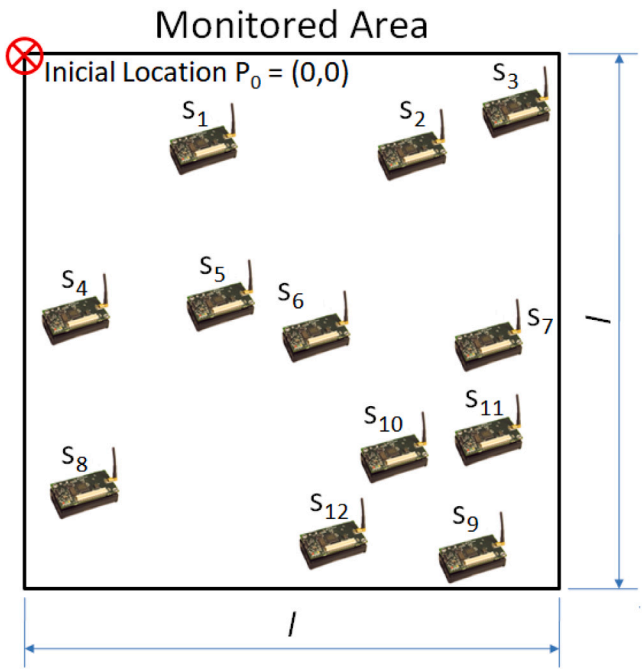


Fig. 2. Example of input: the location of a set of sensor nodes spread within a square monitored area and the amount of data stored in each sensor node memory.

2. Problem statement

This work considers a WSN composed of a set of n sensor nodes $S = \{s_1, s_2, \ldots, s_n\}$ uniform randomly spread over a square monitored area with l meter on the side. We assume that the locations of all nodes are known since the literature presents several works addressing this problem, as shown by the reviews [16,17]. These nodes have sensed environmental data and saved it in their flash memories. They have to send all content of their memories to the sink node. The amount of bytes stored in each sensor node is D. A sensor node needs several data packets to transmit all data since we consider 29 bytes in the payload of each data packet, according [18], and $D \gg 29$. The radio range of the sensor nodes is $r = 60$ m. We consider a quad-copter drone as the mobile sink, which can fly all over the monitored area and can hover anywhere. This device has a radio similar to radios in the sensor nodes. For simplicity, we consider a constant flight speed of $v = 2$ m/s.

The problem analyzed here is **how to find a sequence of locations (named rendezvous point) inside the monitored area, which the mobile sink has to follow to minimize the overall data collecting time**. The mobile sink has to stay a minimal time within the radio range of each sensor node to receive all data stored in their memories. Hence, it has to hover over each of these rendezvous points to have enough time to receive all the data. The mobile sink starts and ends its flight at the Initial Location $P_0 = (0, 0)$. Fig. 2 exemplifies an input of the research problem analyzed here. It presents the square monitored area with l meter of side, the initial location $P_0 = (0, 0)$ and the set of 12 sensor nodes $S = \{S_1, S_2, \ldots, S_{12}\}$. In addition, the problem input also has the amount of data stored in each sensor node memories.

We called the output of this problem as $R = \{T, H, F\}$. A tour $T = (P_0, P_1, \ldots, P_z, P_0)$ is composed of a sequence of rendezvous points within the monitored area, which the drone has to follow and hover over each of them for data collection. The drone always starts and ends its flight at the Initial Location $P_0 = (0, 0)$. Per rendezvous point in T there is a subset of sensor nodes, such that all sensor nodes in the subset H_i have to send data to the drone when it is hovering over the rendezvous point P_i. H is the union of these subsets, consequently $H = H_0 \cup H_1 \cup \cdots \cup H_z$. It is important to mention that we assume

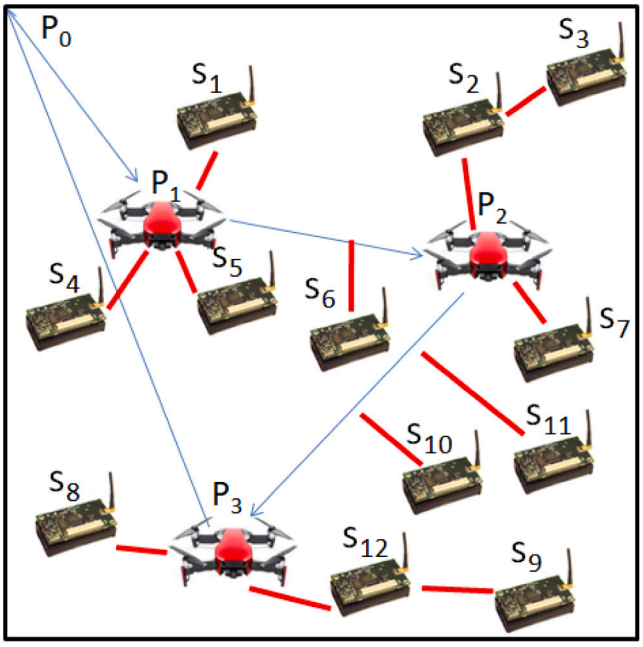


Fig. 3. Example of output: a tour (which is composed of a sequence of rendezvous points) that has to be followed by the drone, a subset of sensor nodes for each rendezvous point (all nodes in a subset have to send data to drone when it is hovering over the same rendezvous point) and a subset of sensor nodes for each par of two consecutive rendezvous points in the tour (all nodes in a subset have to send data to the drone when it is flying between these two rendezvous points).

possible multi-hop communication between any sensor node and the drone over any rendezvous point. Per sequence of two rendezvous points P_i and P_{i+1} in T, there is a subset of sensor nodes F_i that has to send data to the drone during its movement between P_i and P_{i+1}. F is the union of these subsets, such that $F = F_0 \cup F_1 \cup \cdots \cup F_z$. We assume only direct communication between any sensor node and the drone in movement. The drone collects data during its movement to reduce its time hovering, according to Rezende et al. [13]. In this way, the union of H and F contains all sensor nodes, i.e., $S = H \cup F$. Furthermore, the intersection of them is an empty set, that is, $H \cap F = \emptyset$. We assume the drone with low processing power. Hence, a conventional computer has to receive the input, calculate the output and transfer it to the drone. It will follow the tour and gather data from the sensor nodes.

Fig. 3 shows an example of output. The tour $T = (P_0, P_1, P_2, P_3, P_0)$ is the sequence of rendezvous points. The subset of sensor nodes that have to send data to the drone hovering over P_1 is $H_1 = \{S_1, S_4, S_5\}$. The same goes for $H_2 = \{S_2, S_3, S_7\}$ and $H_3 = \{S_8, S_9, S_{12}\}$. Notice that the sensor nodes S_3 and S_9 use multi-hop communication to send their data to the drone. The subset of sensor nodes that has to send data to the drone flying between the rendezvous points P_1 and P_2 is $F_1 = \{S_6\}$, the same for $F_2 = \{S_{10}, S_{11}\}$. In this example $F_0 = \emptyset$ and $F_3 = \emptyset$.

Table 1 summarizes all notations presented in this section.

2.1. Calculating the Drone's flight time

The drone has to follow the tour and hover over each rendezvous point. The total time spent by the drone to start flying in P_0 and return to the same location is called **Overall Time** and receives the notation T_{total}. This is the main metric analyzed here, which we intend to reduce. The following equation calculates this time:

$$T_{total} = T_{flying} + T_{hovering} - T_{moving} \quad (1)$$

T_{flying} is the time spent by the drone in movement, $T_{hovering}$ is the time the drone has to stay hovering over all rendezvous points for data collection and T_{moving} is the time the drone collect data when is moving.

Table 1
Problem notation.

Notation	Description
C	Number of hops used by all sensor nodes to send data to the drone when it is hovering.
D	Amount of data storage in each sensor node memory.
d	Minimal distance among the rendezvous points in the RCL and the rendezvous points in the previous solution.
F	The union of subsets of nodes, such that each subset is composed of the sensor nodes that have to send data to the drone flying between two rendezvous points in the tour.
F_i	Subset of sensor nodes that have to send data to the drone when it is flying between the rendezvous points P_i and P_{i+1}.
H	The union of subsets of nodes, such that each subset is composed of the sensor nodes that have to send data to the drone hovering a specific rendezvous point in the tour.
H_i	Subset of sensor nodes that have to send data to the drone when it is hovering over the rendezvous point P_i.
K	Number of candidates in the restricted candidate list (RCL).
L	Data transfer rate of the links between each par of two sensor nodes or a sensor node and the drone.
l	Side of the monitored area.
$length(T)$	Tour's length, i.e., distance traveled by the drone.
n	Number of sensor nodes in the monitored area.
P_0	Initial and final location of the tour.
P_i	Rendezvous point i, which is a location inside the monitored area and part of a tour.
R	Output of the problem analyzed here, such as $R = \{T, H, F\}$.
r	Radio range of the sensor nodes and mobile sink.
RCL	Restricted Candidate List, subset of K rendezvous points in which one of them will be chosen be part of a new tour.
S	Set of all sensor nodes.
s_i	Sensor node i.
T	Sequence of locations inside the monitored area, called tour, which the drone has to follow and hover over for data collection.
T_i	Tour created in the ith iteration of the heuristics.
T_{flying}	Time spent by the drone to follow the tour.
$T_{hovering}$	Overall time spent by the drone hovering over all rendezvous points for data collection.
T_{moving}	Time spent by the drone flying between two rendezvous points and receiving data at the same time.
T_{total}	Overall time spent by the drone to follow a tour and hover for data collection over each rendezvous point. It is the main metric this work try to minimize.
U	Finite subset of possible rendezvous points inside the monitored area.
u_j	The rendezvous point j.
v	Drone's speed.
w_j	The score of the rendezvous point u_j in the Incremental heuristic, according to Formula (4).
z	Number of rendezvous points in a tour.

T_{flying} is the required time for the drone leaves P_0, flies between each two consecutive rendezvous points, and returns to P_0. Hence, T_{flying} depends on the drone speed v and the length of the tour ($length(T)$). However, the length of the tour depends on the sequence of rendezvous points, which is obtained by solving the well-known Traveling Salesman Problem [12]. The following equation calculates T_{flying}:

$$T_{flying} = \frac{length(T)}{v} \quad (2)$$

$T_{hovering}$ is the sum of the time spent by the drone hovering over each rendezvous point for data collection. This time depends on the link speed (L), the volume of data stored in each sensor node memory (D), and the number of hops (named here as C) in the data paths used

by all sensor nodes to send data to the drone when it is hovering. In this case, we assume possible multi-hop communications. The following equation calculates $T_{hovering}$:

$$T_{hovering} = \frac{(C \times D)}{L} \quad (3)$$

An algorithm proposed by da Silva and Nascimento [14] calculates the value of C. This algorithm receives the location of all sensor nodes in H (those nodes that have to send data when the drone is hovering), the radio range (r) and a tour T. Hence, it calculates the shortest path between each sensor node and the drone over one of the rendezvous points. Then, C is obtained by summing the number of hops used by each sensor node to transmit data to the drone. This algorithm creates a graph considering the sensor nodes and the rendezvous points as vertices and the possible direct communications as edges. It also creates edges connecting the sequence of rendezvous points. The edges connecting two sensor nodes or connecting a sensor node and a rendezvous point receive a weight equal to 1 (one). The edges connecting the rendezvous points receive a weight 0 (zero).

This algorithm considers any rendezvous point as the root and generates the minimal Spanning Tree of this graph. Since the edges connecting the rendezvous points have weights equal to zero, the sensor nodes will be connected to the closest rendezvous point, according to the number of hops. The Prim's or Kruskal's algorithms [12] create the minimal spanning tree. These algorithms have a time complexity of $O(m \log n)$ (where m is the number of edges and n is the number of vertices).

Fig. 4 exemplifies the calculation of C. Fig. 4(A) presents the graph representing a WSN and the rendezvous points that form a tour. The nodes that have to send data to the drone in movement are not considered here. Fig. 4(B) presents the Spanning Tree created by using the previous graph. In this example, the data paths used by the sensor nodes $S_1, S_2, S_3, S_4, S_5, S_7, S_9, S10$ and S_{11} have only one hop, and the data paths used by the sensor nodes S_6, S_8 and S_{12} have two hops, hence $C = 15$.

T_{moving} is the time spent by the drone flying and receiving data. Rezende et al. [13] presented an algorithm to define which subset of sensor nodes has to send data to the drone when it is traveling between each par of consecutive rendezvous points in a tour. Section 4.1.4 presents more details about it. Notice that this algorithm does not change T_{flying}. It only decreases $T_{hovering}$ since it reduces the number of sensor nodes that have to send data to the drone when it is hovering. Fig. 4(C) exemplifies the result of this algorithm. The sensor nodes S_1, S_2, S_4 and S_7 have to send data to the drone in movement.

2.2. Analyzing the problem

This work aims to find a tour T that minimizes the value of T_{total}. Consider a brute force algorithm that always finds the best tour, that is, an algorithm that finds the tour that provides the most minor T_{total}. The first task of this algorithm is to define a finite set of possible rendezvous points. Each monitored area has an infinite number of points inside them. Hence the algorithm must define a finite set of points (called U) to limit possible tours. Then, if U has n possible rendezvous points, there will be $2^n - 1$ possible subsets of U. In other words, there will be $2^n - 1$ possible tours to analyze. This algorithm has to calculate the value of T_{total} for all tours to find the smallest. According to Eq. (1), T_{total} is the sum of T_{flying} and $T_{hovering}$, less T_{moving}. An algorithm to calculate T_{flying} for a given subset of rendezvous points is the same to solve the well-known Travelling Salesman Problem (TSP), which is NP-hard according to Cormen et al. [12]. Moreover, for each possible tour, the value of $T_{hovering}$ varies according to the network topology. It depends on the number of hops used by all sensor nodes to send data to the drone. $T_{hovering}$ tends to be longer when the rendezvous points are far from the sensor nodes, and it tends to be shorter when they are close. Notice that the TSP is part of the problem analyzed here since the rendezvous points that compose the trajectory must also be defined.

According to Rezende et al. [13], tours with many rendezvous points tend to decrease $T_{hovering}$ since it reduces the number of hops in the data paths. Furthermore, the number of sensor nodes sending data to the drone in movement tends to increase on long tours. However, long tours increase the value of T_{flying}. Hence, there is a trade-off between T_{flying} and $T_{hovering}$. These characteristics justify the definition of a heuristic that must find a balance between the flight time and the hovering time.

An important issue that must be clear here is the focus of the research. This work focuses on drone limitations. The proposals try to reduce the overall data gathering time by finding the best rendezvous points and defining when each sensor node has to send data. However, the data gathering problem in wsn is a huge problem with several characteristics that contribute to the network performance. For these characteristics, we considered definitions made in related works. For instance, we assume the channel allocation proposed by da Silva et al. [19], the network energy consumption is not our focus (we analyze it in Section 5.3), and we consider the sensor node geolocations as known according [16,17]. Furthermore, we made some simplifications. For example, we do not consider obstacle avoidance, do not perform many data collection turns, and the drone has a constant speed. If we consider all these characteristics or at least part of them, the number of variables would be impractical.

3. Related works

The literature presents several works that describe techniques to collect data from sensor nodes by mobile sinks. Some define protocols to gather data, which are executed by the sensor nodes to create data paths and forward the sensed data. Other works present algorithms to create tours (or trajectories) that the mobile sink has to follow to collect data. This is the focus of this work. The works that consider sinks following a random or predefined movement pattern are not analyzed here.

He et al. [20] proposed an algorithm to define the trajectory of mobile sinks to reduce the delay for data delivery and extend the network lifetime. This algorithm is based on multi-objective particle swarm optimization. The authors proposed a mechanism to select potential rendezvous points within the sensor nodes' radio range to reduce the mobile sink's trajectory length. Hung et al. [21] also proposed a mobile sink trajectory algorithm to reduce the network energy consumption for data collection. This algorithm randomly defines a percentage of sensor nodes as cluster heads and uses the Dijkstra algorithm to create the mobile sink trajectory. Srivastava et al. [6] followed the same strategy. However, they defined a heuristic based on the Genetic Algorithm metaheuristic to choose 10% of the sensor node positions to form a sequence of rendezvous points followed by the mobile sink. The fitness function defined by them uses the trajectory length, the number of sensor nodes that have to send data to the mobile sink over each rendezvous point, and the sum of hops of all data paths. Preetha et al. [8] proposed a fuzzy-based algorithm for the cluster head selection and mobile sink trajectory definition passing over each cluster head location. The algorithm proposed by Yalçin and Erdem [22] does the same, but the cluster head's selection occurs according to a predefined priority and the energy level. Ghorpade and Vijaykarthik [9] also defined an algorithm to define the mobile sink trajectory according to the location of the cluster heads. The cluster head selection algorithm has a multiple-objective function with several parameters, such as radio strength, connection time, and connectivity.

Hou et al. [23] created a virtual grid to divide the nodes into clusters and select a cluster head. The proposed algorithm defines where the sink has to go after each data gathering in each cluster. The author mentioned that one of the major problems is the mobile sink's slow speed, which demands too much time to collect all data. They consider a sink with no trip distance limitation. Sapre and Mini [5] defined the mobile sink trajectory according to the deployment of relay nodes.

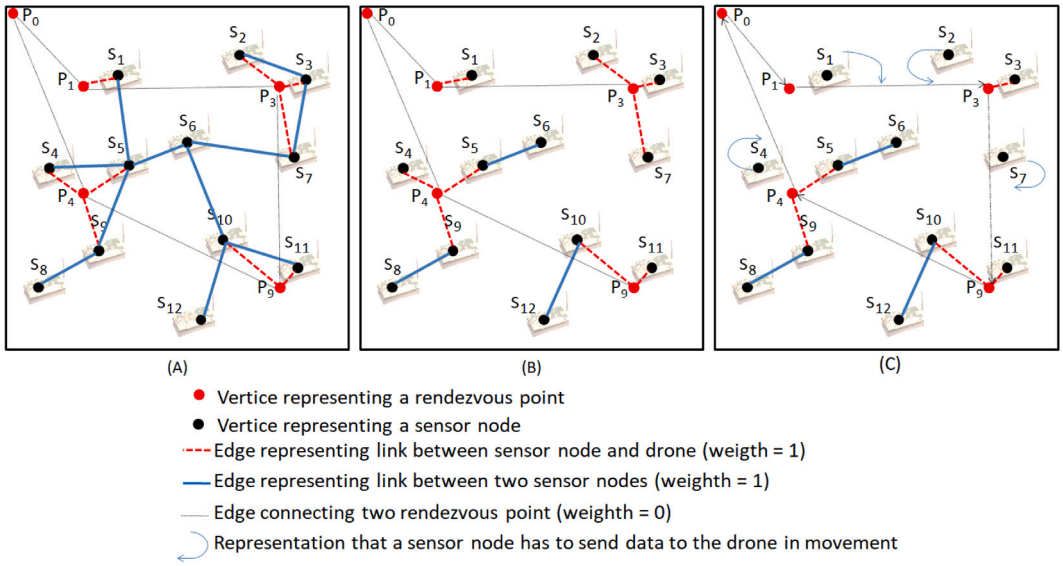


Fig. 4. (A) Graph representing a tour and every possible direct communications between two nodes and between a sensor node and the drone over a rendezvous point. (B) Minimal spanning tree created to calculate the smallest number of hops used by each sensor node to send data to drone. (C) Representation of the sensor nodes (S_1, S_2, S_4 and S_7) that have to send data to the drone in movement.

These nodes have more resources than the other sensor nodes, such as radio range and memory. The WSN is divided into clusters, with the relay nodes as cluster heads. Hence, all nodes forward data to a relay node. The proposed algorithm is based on the metaheuristic Differential Moth Flame Optimization and provides tours in which the mobile sink enter the radio range of all relay node. That is, they consider only direct communication.

Chang et al. [24] proposed an algorithm to define the shortest possible path to be followed by the mobile sink to collect data from the WSN. They consider a square monitored area with 5 kilometers on each side, single-hop communication, and obstacles that the mobile sink must avoid. Anwit et al. [25] proposed two algorithms to define the trajectory of a group of mobile sinks. The first algorithm defines the optimal number of rendezvous points in the trajectory, and the second optimizes the number of mobile sinks to reduce the trajectory's length. Chen and Tang [26] proposed a framework for UAV-assisted data collection. The wsn has a cluster-head selection algorithm. The proposed algorithm calculates the UAV trajectory to reduce this path by considering that the UAV only has to pass over the cluster-head radio range, not exactly over the cluster-heads.

All these works have an important characteristic: each sensor node has only one data packet to transmit to the mobile sink. Hence, these algorithms do not consider a minimum time that the mobile sink has to stay inside the radio range of each sensor node. To the best of our efforts, we found only two groups of articles that consider sensor nodes that need many data packets to deliver to the mobile sink all data stored in their memories. The first group is composed of Li et al. [27,28,29]. They proposed algorithms to define the drone's flight trajectory and data collection protocols for ground sensors. The drone has a constant speed, and it decides the next waypoint of its trajectory on the flight, according to Markov-Chain-based decision algorithms. The authors also proposed data collection protocols that make each sensor node use the transmission channel alone to avoid collisions. However, they focused on reducing the data packet loss and the network energy consumption. They did not evaluate the drone's flight time. Furthermore, these articles consider scenarios where the sensor nodes produce data continuously, and the proposed algorithms do not guarantee that the drone will stay enough time inside the radio range of each sensor node to receive all stored data. For this reason, the drone has to perform its trajectory several times.

Also in this group of articles, Mehto et al. [7] proposed an algorithm to select rendezvous points for data gathering in wsn. They consider non-uniform data generation and buffer limitations in the sensor nodes. The proposed algorithm minimizes the mobile sink trajectory while visiting the rendezvous points. However, the amount of packets to send to the mobile sink is not big. The mobile sink can only pass over the rendezvous points to receive all data. If the amount of data increases, the algorithm does not guarantee that the sensor nodes will send all stored data. Saxena et al. [10] proposed a polynomial algorithm to define the drone trajectory. Their research problem is a version of TSP. Raj et al. [11] proposed a genetic algorithm for the same problem, including obstacle avoidance. These two algorithms do not select the rendezvous points for data collection. The network defines them. They only have to calculate the shortest path. This problem differs from the research problem analyzed here. Our research problem also includes the search for the best rendezvous points. Hence, our problem has higher complexity.

The second group is da Silva and Nascimento [14] and Rezende et al. [13]. They consider sensor nodes with several data packets to transmit. Furthermore, they also consider a quadcopter as a mobile sink with a limited flight time. To the best of our efforts, these works are the only ones considering the mobile sink limitations and proposed algorithm to look for the rendezvous points to compose the tour. Since the second work is an evolution of the first, we consider the algorithms proposed by Rezende et al. [13] as state-of-the-art for data collection in WSN with sensor nodes storing a large volume of data. We have implemented these algorithms and compared them against the GRASP-based algorithms we are proposing here. In the original article, these algorithms are called Incremental-Move and Decremental-Move, but for simplicity, we call them only Incremental and Decremental.

4. The proposed algorithms

This work proposes two GRASP-based heuristics to define the rendezvous points and create tours to reduce the overall time for data collection in WSN by drone. GRASP is a metaheuristic that constructs the solution iteratively. Each iteration is divided into two phases called Construction and Local Search. The Construction phase creates a new solution by adding a new element to the previous solution by using a greedy randomized algorithm. This new element is randomly chosen from a ranked *restricted candidate list* (RCL), which is composed of K

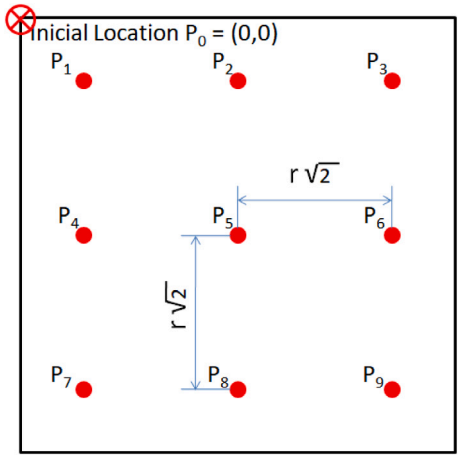


Fig. 5. Subset of possible rendezvous point ($U = \{P_0, P_1, \ldots, P_9\}$) forming a grid. The distance between each par is $r\sqrt{2}$, where r is the radio range.

elements. The Local Search phase improves a solution to find a local minimum. The result of a Grasp-based algorithm is the best overall solution. The number of iterations is defined according to the size of the instance of the analyzed problem [15].

The heuristics proposed here are called Incremental-Grasp and Decremental-Grasp. These heuristics define a limited set of rendezvous points, classify them according to scores and create new solutions in each iteration by adding or removing a rendezvous point from the previous solution. We assume the set of possible rendezvous points forming a grid, such as presented by Fig. 5. The distance between neighbor rendezvous points in the monitored area is $r\sqrt{2}$ (left, right, up, and down). Using this distance, we guarantee that if the drone hovers over all rendezvous points, all points inside this area will be covered by the drone's radio at least once. This means that every sensor node will be able to directly transmit data packets to the drone over at least one rendezvous point. In the following, these heuristics are detailed.

4.1. Incremental-Grasp Heuristic

The Incremental-Grasp Heuristic creates a new solution on each iteration by adding a new rendezvous point to the previous solution, which created in the previous iteration. Algorithm 1 illustrates its operation. This heuristic starts calculating a score for each rendezvous point in U (line 2) and defining both the current solution (line 3) and the best solution (line 4) as empty sets. In every iteration, the heuristic first saves the current solution as the previous (line 6). Then, it performs the Construction phase by creating a new solution with the insertion of a new rendezvous point to the current solution (line 7). The function $NextRendezvousPoint()$ chooses this rendezvous point. It creates the restricted candidate list (RCL) composed of the K rendezvous points with the highest scores. Furthermore, the rendezvous points in the RCL must be at least $2 \times r$ away from the others in the previous solution. Thus, it executes the Local Search phase by calling the function $LocalSearch()$, which looks for a local minimum and saves the result as the current solution (line 8). Section 4.1.3 presents more details about the Local Search phase. The function $DroneMoving(CurrentSolution)$, in the line 9, verifies which sensors will send data when the drone is in movement. It reduces the value of $T_{hovering}$. Section 4.1.4 describes more details about this algorithm. Finally, the function $Evaluate()$ returns which solution provides the smallest T_{total}. This result is saved as the best solution (line 10). The number of iterations is the number of rendezvous points in U. It is essential to mention that every solution, created on each iteration, is evaluated to verify the value of T_{total}, which occurs according to Eq. (1).

4.1.1. Defining the Restricted Candidate List (RCL)

The RCL is composed of K rendezvous points with the highest scores, and it is created on each iteration. One of these rendezvous points is chosen to be added to (by the Incremental-Grasp) or removed from (by the Decremental-Grasp) the previous solution to create the current solution. Section 4.1.2 presents how the Incremental-Grasp heuristic calculates the scores. Moreover, the RCL is created only with the rendezvous points at least a minimal distance (d) away from the rendezvous points in the previous solution. Initially, we defined the minimal distance as $d = 2 \times r$. However, d is divided by 2 if less than K rendezvous points are distant enough from the rendezvous points in the previous solution. This means that this heuristic divides d by 2 until the RCL has K elements.

4.1.2. Defining the scores in the Incremental-Grasp

In the Incremental-Grasp heuristic, each iteration calculates the score of all rendezvous points. This heuristic creates a graph with vertices representing the sensor nodes and the rendezvous points in the previous solution. It also has the edges representing direct communication between two sensor nodes or between sensor nodes and the drone hovering over a rendezvous point. Fig. 4(A) exemplifies this graph. The following equation calculates the score:

$$w_j = \sum_{h=1}^{y} \frac{hop_h}{h} \tag{4}$$

where w_j is the score of the rendezvous point u_j, hop_h is the number of sensor nodes that will send data to the drone for a data path with h hops, and y is the number of hops of the longest data path. Using this formula, the rendezvous points that are close to several sensor nodes receive higher scores.

4.1.3. The local search

Local Search is one of the phases defined by the Grasp metaheuristic. It looks for the best solution in the neighborhood of the solution created on the current iteration. The Local Search in the heuristics proposed here tries to improve the current solution by changing the last added or removed rendezvous point. Each of its four neighbors (up, down, left, and right) replaces this rendezvous point. This strategy creates four new solutions and returns the one that provides the smallest T_{total}. For example, let's consider the rendezvous point P_5 presented by Fig. 5. Assuming that it was the last rendezvous point added to or removed from the current solution, it is replaced by P_2, P_4, P_6, and P_8. Each new tour is evaluated and that one which provides the smallest T_{total} is the result of the Local Search.

4.1.4. Sensor nodes sending data to the drone in movement

After the Local Search, an algorithm proposed by Rezende et al. [13] evaluates the current solution to find which sensor nodes can send data to the drone when it is moving. It looks for a subset of sensor nodes for each pair of rendezvous points in the tour. The sensor nodes in a subset have to send their data when the drone is flying between these rendezvous points. However, the algorithm guarantees that each node will transmit alone to avoid the collision of packets in the drone. This algorithm reduces $T_{hovering}$, and consequently also reduces T_{total} since it decreases the number of sensor nodes that have to send data to the drone hovering.

4.1.5. Example of Incremental-Grasp execution

This section presents an example of the Incremental-Grasp operation. The input is presented by Fig. 6(A), which shows all rendezvous points ($U = \{P_0, P_1, \ldots, P_9\}$), all sensor nodes ($S = \{s_1, s_2, \ldots, s_6\}$) and the connections among them. This heuristic starts calculating the score of each rendezvous point, such as presented by Table 2.

Fig. 6(B) presents all data paths and the tour created after the first iteration. We consider the RCL with $k = 2$ elements, hence it was composed of P_5 and P_6, which had the highest scores ($w_5 = 4.33$ and

Table 2
Calculating the score of each rendezvous point for Incremental-Grasp example.

Score	Value	Calculus
w_1	0	
w_2	0	
w_3	2.45	$\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\frac{1}{6}$
w_4	2.42	$\frac{1}{1}+\frac{2}{2}+\frac{2}{3}+\frac{1}{4}$
w_5	4.33	$\frac{3}{1}+\frac{2}{2}+\frac{1}{3}$
w_6	3.67	$\frac{2}{1}+\frac{2}{2}+\frac{2}{3}$
w_7	0	
w_8	2.45	$\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\frac{1}{6}$
w_9	3.28	$\frac{2}{1}+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}$

Algorithm 1 Incremental-Grasp Heuristic

```
1:  procedure IncrementalGrasp(S, U)
2:      CalculateScores(S, U)
3:      CurrentSolution ⟵ ∅
4:      BestSolution ⟵ ∅
5:      for i = 0 to i < length(U) do
6:          PreviousSolution ⟵ CurrentSolution
7:          CurrentSolution ⟵ CurrentSolution + AddRendezvousPoint()
8:          CurrentSolution ⟵ LocalSearch(CurrentSolution)
9:          CurrentSolution ⟵ DroneMoving(CurrentSolution)
10:         BestSolution ⟵ Evaluate(CurrentSolution, BestSolution)
11:     end for
12:     return BestSolution
13: end procedure
```

$w_6 = 3.67$). The rendezvous point P_5 was randomly chosen to create the first tour. After this, the Local Search was performed to find the local minimum. It replaced P_5 for P_2, P_4, P_6 and P_8. Since the tour composed by P_0 and P_5 provided the smallest T_{total}, the Local Search did not change the current answer. Then, the heuristic found S_2 as a node able to send data when the drone was flying between P_0 and P_1.

Fig. 6(C) presents the tour created after the second iteration. P_9 was randomly chosen from the RCL composed of P_3 and P_9. Notice that P_4, P_6, and P_8 could not be part of the RCL because they are less than $d = 2 \times r$ close to P_5. In this iteration, the local search replaced P_9 by P_6 and P_8, but this replacement does not decrease T_{total}. The sensor node S_6 was chosen to send data to the drone when it is flying between P_5 and P_9.

Fig. 6(D) presents the tour created in the third iteration by adding P_6. This rendezvous point was randomly chosen from the RCL composed of P_4 and P_6. Notice that the minimal distance (d) must be divided by 2 ($d = \frac{2\times r}{2} = r$) to create the RCL with two elements. The Local Search replaced P_6 with P_3, but it did not create a better tour. Since P_5 and P_9 were part of the current solution, the Local Search did not consider them. The algorithm proceeded adding to the tour all the other rendezvous points. However, for simplicity, we do not present it. The result is the best solution so far.

4.2. Decremental-Grasp Heuristic

The Decremental-Grasp Heuristic creates a new solution on each iteration by removing a rendezvous point from the tour created in the previous solution. Algorithm 2 illustrates this heuristics. We also assume all the possible rendezvous points forming a grid, such as presented by Fig. 5. This heuristic starts creating the first tour composed of all rendezvous points that have at least one sensor node connected to it, i.e., all locations where the drone has to hover to receive data from at least a sensor node (line 2). The first tour is saved as the best solution so far (line 3). At the begin of each iteration, the Decremental-Grasp heuristic creates the score for each sensor node by calling the function *ListRemovalImpact(S, U)* (line 5). The score used by the Decremental-Grasp is called *Impact of Removal*, which is better explained in Section 4.2.1. Then, it saves the current solution as the previous solution (line 6). After this, it performs the Construction phase by executing the function *RemoveRendezvousPoint()*, which choose the rendezvous point that will be removed from the current solution (line 7). This function creates the RCL with the K rendezvous points in the current solution with the smallest Impact of Removal. Then, it randomly chooses one of them to be removed from the previous solution. Consequently, it creates the new current solution. The Local Search phase is performed by calling the function *LocalSearch()* to find the local minimum (line 8). Section 4.1.3 presents more details of this phase, which performs in the same way as the Local Search of the Incremental-Grasp. The function $DroneMoving(CurrentSolution)$, in the line 9, verifies which sensors will send data when the drone is moving. This algorithm is described Section 4.1.4. Finally, the current solution is compared with the best solution so far by the function *Evaluate* and the best of them is saved (line 10).

Notice that, according to Section 2.1, in each iteration, this heuristic also creates a graph, which has the sensor nodes and the rendezvous points as vertices. The edges are the possible direct communication between each par of sensor nodes (respecting the radio range) or between a sensor node and the drone hovering over a rendezvous point. The heuristic creates the spanning tree from this graph, which has the smallest data path (in hops) between each sensor node and a rendezvous point.

Algorithm 2 Decremental-Grasp Heuristic

```
1:  procedure DecrementalGrasp(S, U)
2:      CurrentSolution ⟵ U − Disconnected(U)
3:      BestSolution ⟵ CurrentSolution
4:      for i = 1 toi < len(U) do
5:          ListRemovalImpact(S, U)
6:          PreviousSolution ⟵ CurrentSolution
7:          CurrentSolution ⟵ CurrentSolution − RemoveRendezvousPoint()
8:          CurrentSolution ⟵ LocalSearch(CurrentSolution)
9:          CurrentSolution ⟵ DroneMoving(CurrentSolution)
10:         BestSolution ⟵ Evaluate(CurrentSolution, BestSolution)
11:     end for
12:     return BestSolution
13: end procedure
```

4.2.1. Calculating the Impact of Removal

The *Impact of Removal* is the score used by the Decremental-Grasp heuristic to classify the rendezvous points. The RCL is composed of the K rendezvous points with the smallest Impact of Removal. This heuristic defines the same graph created by the previous heuristics and its spanning tree to create the shortest path between every sensor node and the drone hovering over one of the rendezvous points in the tour. However, it removes a rendezvous point on each iteration. Consequently, the edges connected to this rendezvous point also have to be removed, and new edges have to be added to define data paths

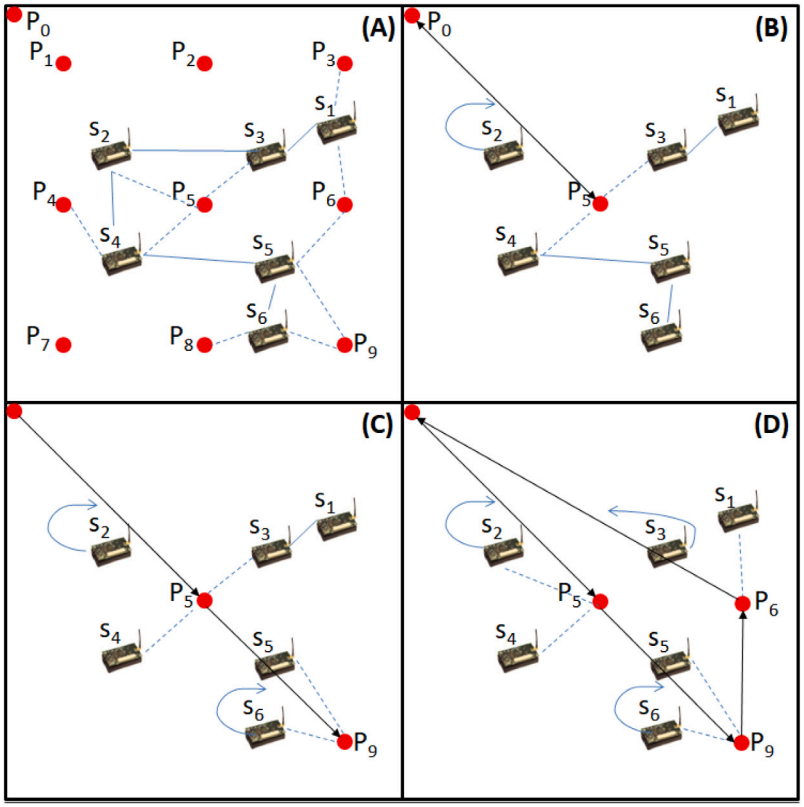


Fig. 6. Example of the Incremental-Grasp operation: (A) Problem input, a set of 9 possible rendezvous points, 6 sensor nodes, and the connection among them. (B) The tour that was created after the first iteration; the RCL was composed of P_5 and P_6, since they had the highest scores; P_5 was randomly chosen to create the first solution; S_2 was chosen to send data during the drone movement. (C) The tour that was created after the second iteration; P_9 was randomly chosen to be added to the tour since the RCL was composed of P_3 and P_9; The rendezvous points P_6, P_4 and P_8 were not considered here because they were less than the minimal distance $d = 2 \times r$ close to the rendezvous points in the previous tour; S_6 was chosen to send data during the drone movement. (D) The tour that was created after the third iteration; the minimal distance was divided by 2, hence the RCL was composed of P_4 and P_6; P_6 was randomly chosen; S_3 was chosen to send data during the drone movement.

for all sensor nodes. The Impact of Removal is the difference between the number of hops added by the number of hops removed.

For example, given a tour with the rendezvous point u_j and only a sensor node s_i directly connected to it (one hop). The sensor node s_i can also send data to the drone hovering over another rendezvous point u_e in the same tour. However, it uses a data path with two hops. In this case, the Impact of Removal of u_j is one since a data path with one hop is removed and another with two hops is added.

4.2.2. Example of Decremental-Grasp execution

This subsection exemplifies the operation of the proposed Decremental-Grasp heuristic. Fig. 7(A) presents an input, which is composed of a monitored area, a set of sensor nodes $S = \{s_1, s_2, \ldots, s_6\}$ and a set of possible rendezvous points $U = \{P_0, P_1, \ldots, P_9\}$. In the beginning, the heuristic creates an initial tour $T_0 = (P_0, P_4, P_5, P_9, P_3, P_0)$ with the rendezvous points where the drone will hover over for data collection.

In the first iteration, the heuristic calculates the Impact of Removal for each rendezvous point, according to Section 4.2.1. For example, the rendezvous point P_5 has the Impact of Removal $w_5 = 2$. If P_5 was removed, the data path of the sensor nodes S_2 and S_3 (both with one hop) have to be replaced. S_2 will send data to s_4, that will relay all data packets to the drone over P_4 (two hops). s_3 has to send their data to s_1, which will relay to P_3 (two hops). Hence, two hops will be removed and four hops will be added, i.e., $w_5 = 4 - 2 = 2$. The Impact of Removal of all rendezvous points are: $w_3 = 1$, $w_4 = 0$, $w_5 = 2$, $w_6 = 0$ and $w_9 = 1$. Hence, the RCL is composed of P_4 and P_6, which are the rendezvous points with the smallest scores. In this example, P_4 is chosen to be removed from the tour. Furthermore, the function $DroneMoving()$ verifies that S_1, S_2 and S_5 will send data while the drone is in movement. The result of the first iteration is presented by Fig. 7(C).

The scores in the second iteration are $w_3 = 1$, $w_5 = 3$, $w_6 = 0$ and $w_9 = 1$. The RCL is composed of $w_3 = 1$ and $w_6 = 0$, hence P_3 is randomly chosen to be removed from the previous solution. The function $DroneMoving()$ inform that s_1, s_2 and s_6 can send data to the drone in movement. Fig. 7(D) presents the result of the second iteration. After this, the heuristic continues removing rendezvous points from the tour. It stops only when the tour is composed of P_0.

5. Computational results

This section compares the heuristics Incremental-Grasp and Decremental-Grasp proposed here against the heuristics Incremental and Decremental proposed by Rezende et al. [13], which represent state of the art for this type of scenario.

5.1. Scenario description

This subsection describes the four scenarios used by the experiments. Since the algorithms proposed in [13] represent state of the art for this type of scenario, we implemented the Incremental and the Decremental heuristics proposed in this work. Furthermore, we performed the experiments in the same scenarios. We created the four heuristics in Java 8, 64 bits. The computer used to run the experiments has an Intel Core I7-8565U, 1.8 GHz, and 8 GiB of RAM. However, it is essential to mention that the metric execution time is not analyzed, and every computer would present the same results.

We consider a drone with hovering capability as the mobile sink, such as a quadcopter. It can fly and hover over any point in the

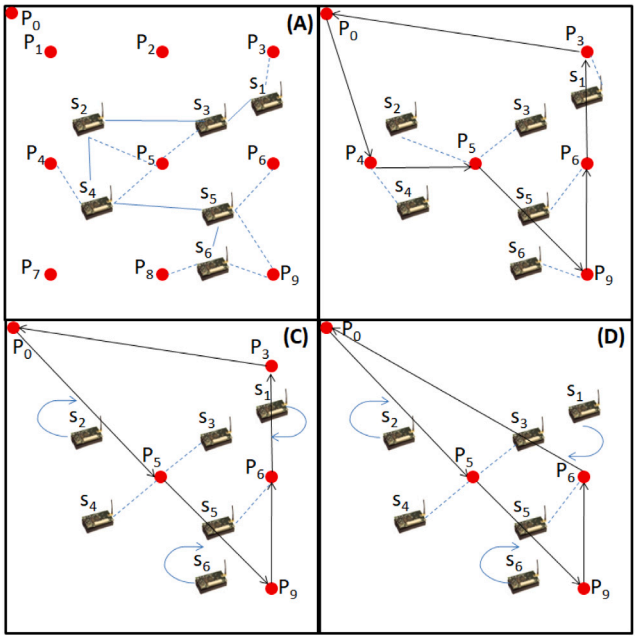


Fig. 7. Example of the Decremental-Grasp operation: (A) Input of the problem, a set of 9 rendezvous points, 6 sensor nodes, and the connection among them. (B) The initial tour created with all rendezvous points that have sensor nodes connected to them. (C) The tour created after the first iteration; P_4 was randomly chosen from the RCL to be removed from the tour and the sensor nodes s_1, s_2 and s_6 will send data during the drone movement. (D) The tour created after the second iteration; P_3 was removed and s_1, s_2 and s_6 continues sending data during the drone movement.

Table 3
Characteristics of the scenarios.

Scenario	Monitored area (m^2)	Number of sensor nodes	Drone speed (m/s)	Data in each sensor node (kb)
1	200	30	2	20 to 120
2	400	150	2	20 to 120
3	400	100 to 250	2	60
4	400	150	0.5 to 3.0	60

monitored area. The mobile sink has a radio like the sensor nodes, with the same range (r = 60 m). The transmission rate of every link of the WSN is 20 kbps, the same as the drone. Each node transmits separately to avoid packet collision, according to da Silva et al. [19]. In all scenarios, U is composed of rendezvous points with $r \times \sqrt{2} = 84$ m of distance between two of them (up, down, left, and right). In this way, every point inside the monitored area is less than 60 m far from a rendezvous point. The time to propagate queries is not considered here. The drone moves at a constant speed (v) and collects data when it is hovering and when it is moving.

Table 3 summarizes all characteristics of the four scenarios.

We used 35 different network topologies in all scenarios presented by Table 3. In Scenario 1, there is a small square monitored area with 200 m of side and 30 fixed sensor nodes. The drone speed is 2 m/s, and the data stored in each sensor node varies in the set $\{20, 40, 60, 80, 100, 120\}$ kb. Since there are 35 WSN topologies and 6 different amounts of drone storage, there are 210 instances in Scenario 1.

Scenarios 2 to 4 consider a larger monitored area and a larger number of sensor nodes to evaluate the heuristics' performance in WSN with data routes longer than in the first scenario. All of them have a square monitored area with 400 m of side. As in the previous scenario, Scenario 2 also varies the amount of data stored in each sensor node memory from 20 to 120 kb. In Scenario 3, the number of sensor nodes varies from 100 to 250 with step 50, i.e., $100, 150, 200, 250$. In Scenario 4, the drone speed varies from 0.5 to 3.0 m/s. i.e., $0.5, 1.0, 1.5, 2.0, 2.5, 3.0$ m/s. So, there are 210, 180, and 210 instances in Scenarios 2, 3, and 4, respectively.

Table 4
Comparison between Incremental and Incremental-GRASP in relation to the T_{total} value.

Instance	# Data (kb)	Incremental	Incremental-GRASP		
			Best	Avg.	RPD (%)
1	20	155.17	149.12	151.68	−2.24
2	40	208.27	201.67	206.57	−0.81
3	60	249.52	243.31	248.29	−0.49
4	80	287.14	282.39	284.77	−0.82
5	100	321.82	312.51	320.34	−0.45
6	120	351.55	349.61	350.45	−0.31

Table 5
Comparison between Decremental and Decremental-GRASP concerning the T_{total} value.

Instance	# Data (kb)	Decremental	Decremental-GRASP		
			Best	Avg.	RPD (%)
1	20	245.72	241.24	243.95	−0.72
2	40	269.91	244.77	253.57	−6.05
3	60	284.14	234.02	255.12	−10.21
4	80	295.87	220.11	246.88	−16.55
5	100	307.35	230.37	246.44	−19.81
6	120	317.74	207.38	243.15	−23.47

5.2. Experiments

In this section, we compare the results of Incremental and Decremental heuristics proposed by Rezende et al. [13] against Incremental-GRASP and Decremental-GRASP heuristics proposed here. Initially, we applied the *irace* package [30] for calibrating the parameter K of the proposed GRASP-based heuristics. It determines the number of rendezvous points in the restricted candidate list (RCL) at each iteration of the Incremental-GRASP and Decremental-GRASP. We tested the values $k \in \{2, 3, 4\}$ in a subset of 18 instances of the Scenario 1 varying the amount of data stored in each sensor node. The *irace* package returned that the best value is $k = 2$. In the next two Sections 5.2.1 and 5.2.2, we tested these methods in small and large monitored areas, respectively.

5.2.1. Small monitored area

In this subsection, we report the results of the Incremental-GRASP and the Decremental-GRASP heuristics in Scenario 1 that involves small monitored areas. Table 4 reports the results found for 30 runs of the Incremental-GRASP heuristic. The first column indicates the number of the instance, the second shows the amount of data stored in each sensor node in kb, and the third column shows the values of T_{total} returned by the Incremental. The following three columns report, respectively, the best and the average value of T_{total} presented by the Incremental-GRASP heuristic and the Relative Percentage Deviation (RPD).

The RPD is evaluated according to Eq. (5):

$$RPD_i = \frac{(T_{total})_i^{Method} - (T_{total})_i^{\star}}{(T_{total})_i^{\star}} \quad (5)$$

in which $(T_{total})_i^{Method}$ is the average T_{total} demanded by the method (Incremental-GRASP or Decremental-GRASP) in 30 executions and $(T_{total})_i^{\star}$ is the value found by the Incremental (or Decremental) for the instance i.

In turn, Table 5 reports the results of the Decremental-GRASP heuristic.

As can be seen from Tables 4 and 5, the proposed GRASP-based heuristics outperformed the performance of the Incremental and Decremental. The improvement is more significant for the Decremental-GRASP method. This behavior of the Decremental-GRASP method occurs due to the following reason. Decremental chooses the hovering

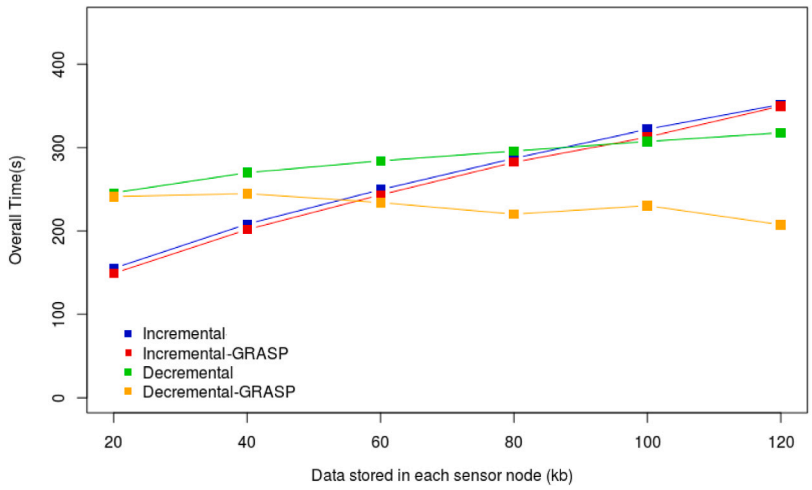


Fig. 8. Best results generated by the methods concerning the overall data gathering time (T_{total}) in a small monitored area (Scenario 1).

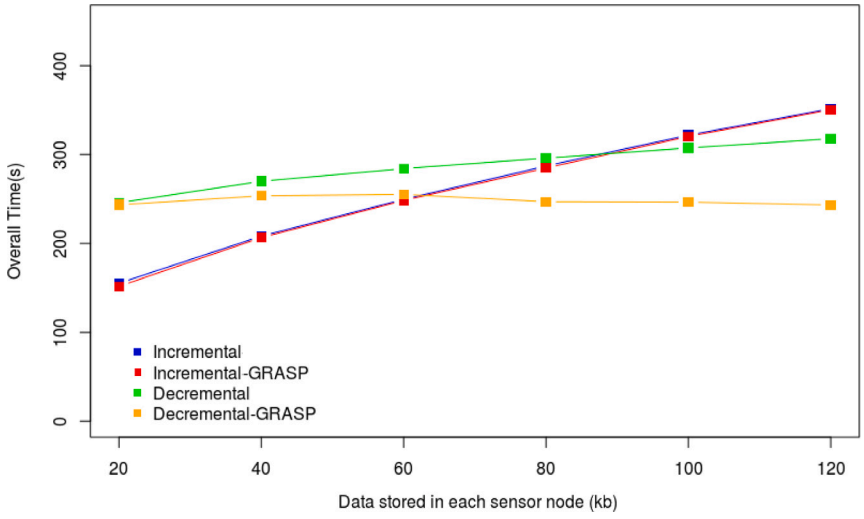


Fig. 9. Average results generated by the methods concerning the overall data gathering time (T_{total}) in a small monitored area (Scenario 1).

point with the smallest Impact of Removal to be removed from the tour at each iteration. In turn, at each iteration, Decremental-GRASP removes from the tour a hovering point between the K rendezvous points with the smallest Impact of Removal. As Decremental tends to have larger tours than Incremental, the value of $K = 2$ is relatively small concerning the tour's size. As a result, the rendezvous point removed has a low Impact of Removal; consequently, the collection time $T_{hovering}$ is less affected.

On the other hand, Decremental-GRASP tends to have tours composed of more rendezvous points than Incremental-GRASP due to its strategy of reducing the tour's size in each iteration. This fact occurs mainly when the amount of data stored in the sensor nodes' memories is greater. In this situation, the more rendezvous points, the smaller the data routes' size and, consequently, the shorter the collection time. Besides, with larger tours, more sensor nodes can send their data to the drone in movement.

Fig. 8 illustrates the heuristics' comparison results, considering the best results produced by them. In turn, Fig. 9 illustrates the heuristics' comparison results, considering the average results generated by them. Notice that, Decremental-GRASP outperforms all other methods when the amount of data stored in the sensor nodes is greater than or equal to 60 kb.

Table 6
Comparison between Incremental and Incremental-GRASP in relation to the T_{total} value for the Scenario 2.

Instance	# Data (kb)	Incremental	Incremental-GRASP		
			Best	Avg.	RPD (%)
1	20	723.14	704.56	720.67	−0.34
2	40	1008.39	987.78	1001.13	−0.71
3	60	1178.64	1146.87	1156.21	−1.9
4	80	1297.71	1246.89	1256.43	−3.18
5	100	1452.67	1398.98	1403.71	−3.37
6	120	1593.93	1556.98	1545.26	−3.05

5.2.2. Large monitored area

Similar to Section 5.2.1, Tables 6–11 report the results found by the Incremental-GRASP and Decremental-GRASP heuristics within a large monitored area (Scenarios 2, 3 and 4) concerning the T_{total}. As before, each heuristic was executed 30 times.

Like in the previous subsection, the GRASP-based heuristics presented better performance than the Incremental and Decremental heuristics. Figs. 10, 12 and 14 illustrate the heuristics' comparison results, considering the best results produced by them. In turn, Figs. 11, 13, and 15 illustrate the heuristics' comparison results, considering the average results generated by them.

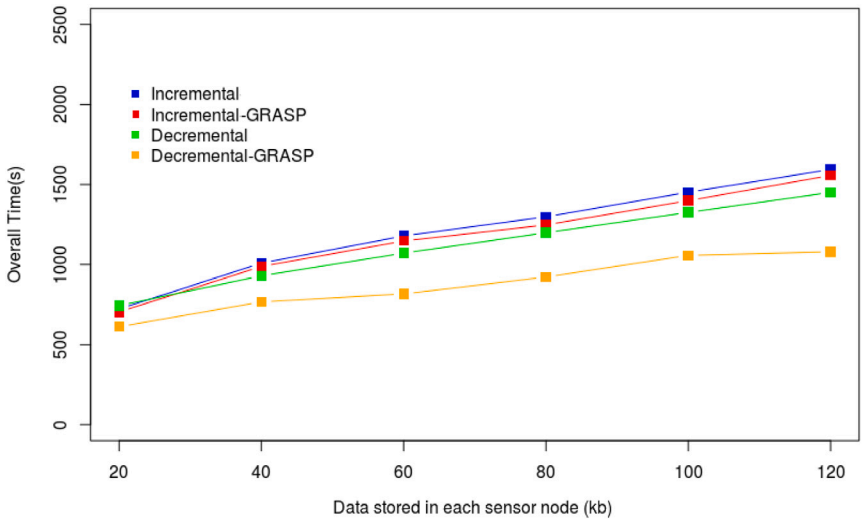


Fig. 10. Best results generated by the methods concerning the overall data gathering time (T_{total}) in a large monitored area (Scenario 2).

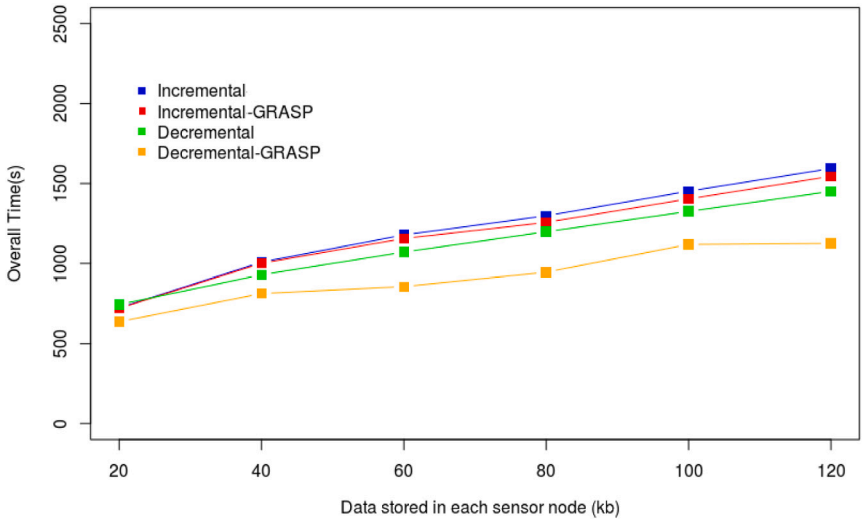


Fig. 11. Average results generated by the methods concerning the overall data gathering time (T_{total}) in a large monitored area (Scenario 2).

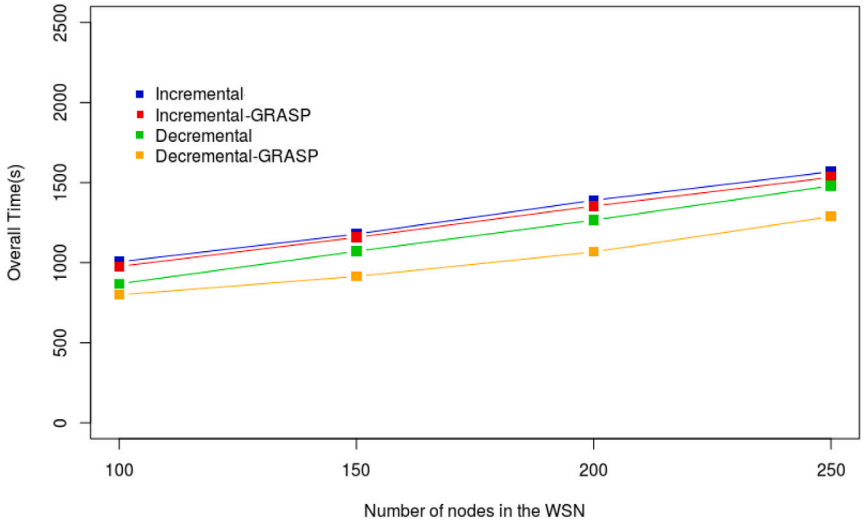


Fig. 12. Best results generated by the heuristics according to the number of nodes in the WSN: large monitored area (Scenario 3).

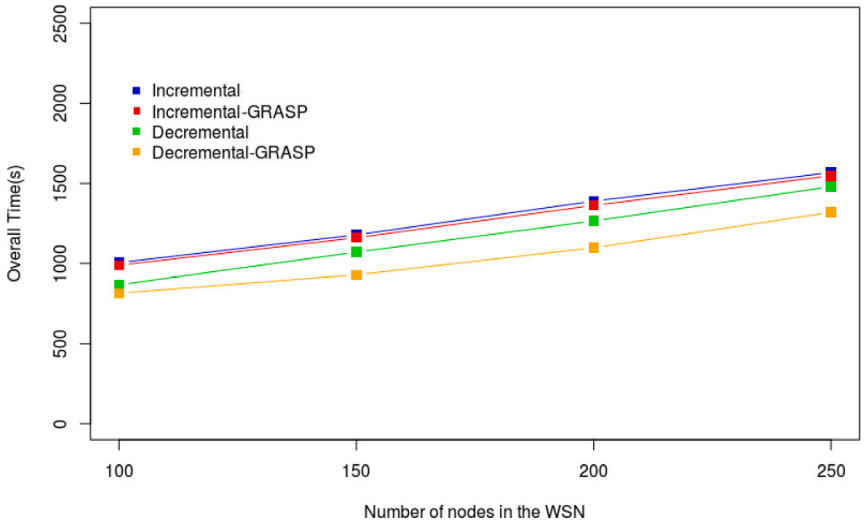


Fig. 13. Average results generated by the methods according to the number of nodes in the WSN: large monitored area (Scenario 3).

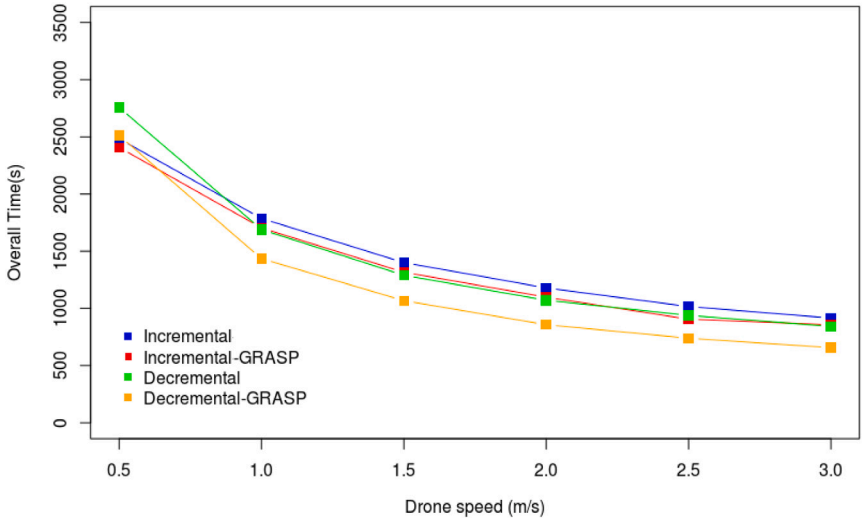


Fig. 14. Best results generated by the methods according to the drone speed in a large monitored area (Scenario 4).

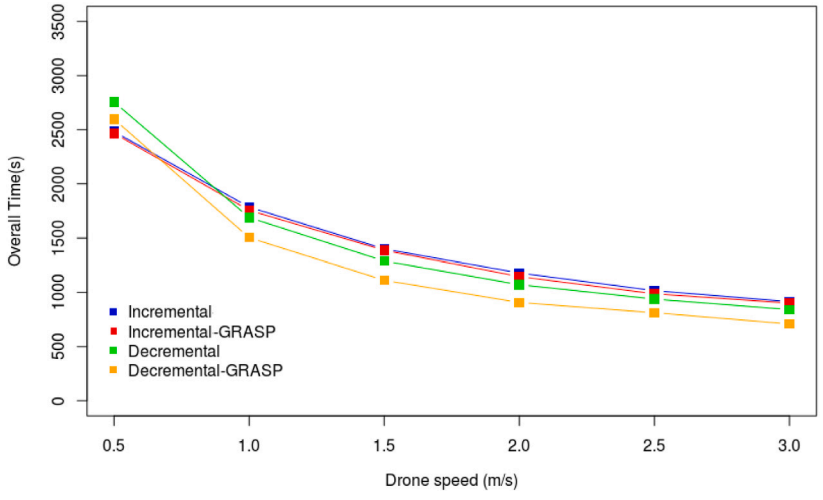


Fig. 15. Average results generated by the methods according to the drone speed in a large monitored area (Scenario 4).

Table 7
Comparison between Decremental and Decremental-GRASP in relation to the T_{total} value for the Scenario 2.

Instance	# Data (kb)	Decremental	Decremental-GRASP		
			Best	Avg.	RPD (%)
1	20	744.41	611.98	636.9	−14.43
2	40	929.49	767.45	813.11	−12.52
3	60	1070.94	817.56	856.3	−20.04
4	80	1198.6	921.87	945.67	−21.10
5	100	1325.16	1056.78	1118.74	−15.57
6	120	1450.66	1078.98	1125.16	−22.43

Table 8
Comparison between Incremental and Incremental-GRASP according to to the number of nodes in the WSN.

Instance	Number of sensor nodes	Incremental	Incremental-GRASP		
			Best	Avg.	RPD (%)
1	100	1005.21	976.22	988.56	−1.65
2	150	1178.64	1157.94	1160.78	−1.51
3	200	1389.43	1353.45	1362.64	−1.92
4	250	1568.17	1532.87	1547.97	−1.28

Table 9
Comparison between Decremental and Decremental-GRASP according to the number of nodes in the WSN.

Instance	Number of sensor nodes	Decremental	Decremental-GRASP		
			Best	Avg.	RPD (%)
1	100	867.27	798.98	816.19	−5.88
2	150	1070.94	913.76	929.30	−13.22
3	200	1265.16	1067.15	1098.10	−13.2
4	250	1479.89	1287.49	1319.36	−10.84

Table 10
Comparison between Incremental and Incremental-GRASP in relation to drone speed (m/s).

Instance	Drone speed (m/s)	Incremental	Incremental-GRASP		
			Best	Avg.	RPD (%)
1	0.5	2483.17	2407.23	2467.11	−0.64
2	1.0	1785.97	1703.34	1756.17	−1.66
3	1.5	1400.19	1315.56	1387.70	−0.89
4	2.0	1178.64	1098.24	1145.21	−2.83
5	2.5	1015.83	904.84	986.87	−2.85
6	3.0	915.06	856.56	901.43	−1.48

Table 11
Comparison between Decremental and Decremental-GRASP in relation to drone speed (m/s).

Instance	Drone speed (m/s)	Decremental	Decremental-GRASP		
			Best	Avg.	RPD (%)
1	0.5	2756.83	2512.238	2598.14	−5.75
2	1.0	1689.49	1434.56	1503.86	−10.98
3	1.5	1288.91	1067.44	1111.33	−13.62
4	2.0	1070.94	856.55	907.30	−15.28
5	2.5	938.61	1056.78	812.71	−13.41
6	3.0	841.73	1078.98	708.64	−15.81

As shown in all the previous figures of this subsection (Figs. 10–15), Decremental-GRASP outperforms all other methods within a large monitored area.

5.2.3. Statistical analysis

This subsection analyzes whether there is a statistical difference in the results presented here. We initially applied the Shapiro–Wilk test [31] on each sample that generated a mean result. According to this test, we verify that all samples have a normal distribution. So, we applied the paired t-test [32] on all topologies to the method pairs, i.e., Incremental × Incremental-GRASP, Decremental × Decremental-GRASP, and Incremental-GRASP × Decremental-GRASP. Tables 12–15 report the observed values for t in these comparisons. As before, column 'Methods' represents the pair of methods being compared, and column 'Data stored in each sensor node' informs the amount of data stored in each sensor node, in kb. We set the significance level to 0.05 (5%). At this significance level, the critical value for t is 2.045.

Table 12
Paired t-test results involving the proposed methods in Scenario 1.

Methods	Data stored in each sensor node					
	20	40	60	80	100	120
Incremental × Incremental-GRASP	0.98	0.58	0.68	0.93	0.09	0.79
Decremental × Decremental-GRASP	1.96	2.45	3.12	5.00	6.03	4.11
Incremental-GRASP × Decremental-GRASP	7.66	6.99	2.43	2.80	5.32	5.52

Table 13
Paired t-test results involving the proposed methods in Scenario 2.

Methods	Data stored in each sensor node					
	20	40	60	80	100	120
Incremental × Incremental-GRASP	0.45	0.67	0.98	1.54	2.01	2.04
Decremental × Decremental-GRASP	2.31	2.57	4.12	5.78	5.04	6.13
Incremental-GRASP × Decremental-GRASP	2.76	3.96	5.85	5.79	5.87	6.92

Table 14
Paired t-test results involving the proposed methods in Scenario 3.

Methods	Number of sensor nodes			
	100	150	200	250
Incremental × Incremental-GRASP	0.73	0.81	0.93	1.29
Decremental × Decremental-GRASP	2.05	2.94	3.11	3.09
Incremental-GRASP × Decremental-GRASP	3.21	3.83	3.87	3.94

Table 15
Paired t-test results involving the proposed methods in Scenario 4.

Methods	Drone speed (m/s)					
	0.5	1.0	1.5	2.0	2.5	3.0
Incremental × Incremental-GRASP	0.49	0.58	0.69	0.92	0.95	0.81
Decremental × Decremental-GRASP	2.06	2.31	2.42	2.49	2.35	2.28
Incremental-GRASP × Decremental-GRASP	1.93	3.45	3.92	3.67	3.23	3.01

We note that there is no proven statistical difference between the Incremental and Incremental-GRASP since all the values observed for t in line 1 of Tables 12, 13, 14, and 15 are less than the critical t value.

On the other hand, we can verify a statistical difference between the Decremental and Decremental-GRASP methods in instances whose sensor nodes have 40 kb or more of stored data in the Scenario 1. We can make this statement because the observed value for t in line 2 of Table 12 is greater than the critical value for t when sensor nodes have 40 kb or more of stored data. In the Scenarios 2, 3, and 4, we can verify a statistical difference between the Decremental and Decremental-GRASP methods in all instances, as shown in Tables 13, 14, and 15.

In turn, we prove a statistical difference with a 95% confidence interval between the Incremental-GRASP and the Decremental-GRASP methods. This statement can be affirmed since all the values observed for t in line 3 of Tables 12, 13, 14, and 15 are greater than the critical t value. Thus, given the results of Tables 12 to 15, involving both small and large monitored areas, we can statistically assure that the Decremental-GRASP method is better than all the other methods in the Scenarios 1, 2, 3, and 4.

5.3. Network energy consumption analysis

This work focused on drone limitations. Hence the main metric analyzed here is the overall data gathering time. We do not examine

the network energy consumption. To perform this analysis, we would create or simulate the network infrastructure. Furthermore, we also would have to define the network routing protocol to receive queries from the drone and send back the required data. da Silva et al. [19] proposed algorithms for sensor nodes to answer queries issued by drones. They verified that reducing the overall data gathering time also reduces energy consumption. The overall data gathering time is closely related to the number of hops used to receive data from the sensor nodes. Hence, heuristics that presents shorter data gathering times tend to provide smaller energy consumption.

6. Conclusion and future works

This work proposed two GRASP-based heuristics to solve the problem of finding a tour to be followed by a mobile sink for data collection in WSN to minimize the overall data gathering time. The heuristics are called Incremental-GRASP and Decremental-GRASP. We considered a quadcopter drone as a mobile sink, which can fly and hover over any point inside the monitored area. However, it has a flight time limitation. Furthermore, we consider sensor nodes storing a large volume of data to be collected by the drone. Consequently, the proposed heuristic must guarantee that the drone will stay at least a minimal time inside the radio range of the sensor nodes to receive all data. We compared the proposed heuristics against the heuristics described by Rezende et al. [13], here called Incremental and Decremental. These algorithms represent state of art for this type of scenario.

In the simulated experiments, the proposed Incremental-Grasp outperformed the Incremental heuristic in practically all experiments. The same goes for Decremental-Grasp and Decremental. Moreover, Decremental-Grasp outperformed all the other heuristics in the majority of the scenarios. Its good performance has to be highlighted in scenarios where the sensor nodes have the largest volume of data to transmit to the drone. This performance happens because Decremental-Grasp (also Decremental) tends to provide more long tours than Incremental. This type of tour enables more sensor nodes to send data during the drone movement. Finally, notice that the Grasp-based heuristics provided better results than the greed heuristics found in the literature.

In future works, we intend to improve and evaluate the performance of the proposed heuristics in environments with more complex requirements, such as Large-scale Wireless Sensor Networks (LSWSN). These networks are composed of sensor nodes spread on a large region. Since the density of nodes is low, they can be organized into groups so that some groups have no connection with others. In this situation, the mobile sink must pass over each group to collect all available data. Furthermore, we intend to change the proposed heuristics to define tours for drones in disaster situations, such as after earthquakes and flooding. These situations can have some nodes' mobility and different types of nodes.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

The authors would like to thank the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* - Brazil (CAPES) - Finance Code 001, *Fundação de Amparo à Pesquisa do Estado de Minas Gerais* (FAPEMIG, grant PPM/CEX/676-17), *Conselho Nacional de Desenvolvimento Científico e Tecnológico* (CNPq, grant 303266/2019-8), *Universidade Federal de São João Del Rei* (UFSJ), and *Universidade Federal de Ouro Preto* (UFOP), for supporting the development of the present study.

References

[1] J. Yick, B. Mukherjee, D. Ghosal, Wireless sensor network survey, Comput. Netw. 52 (12) (2008) 2292–2330, http://dx.doi.org/10.1016/j.comnet.2008.04.002, URL: https://www.sciencedirect.com/science/article/pii/S1389128608001254.

[2] R.E. Mohamed, A.I. Saleh, M. Abdelrazzak, A.S. Samra, Survey on wireless sensor network applications and energy efficient routing protocols, Wirel. Pers. Commun. 101 (2) (2018) 1019–1055, http://dx.doi.org/10.1007/s11277-018-5747-9.

[3] R. Balamurali, K. Kathiravan, A survey on mitigating hotspot problems in wireless sensor networks, Int. J. Appl. Eng. Res. 10 (2015) 5913–5921.

[4] J. Chang, J. Jeng, Y. Sheu, Z. Jian, W. Chang, An efficient data collection path planning scheme for wireless sensor networks with mobile sinks, EURASIP J. Wirel. Commun. Netw. 2020 (1) (2020) 257, http://dx.doi.org/10.1186/s13638-020-01873-4.

[5] S. Sapre, S. Mini, A differential moth flame optimization algorithm for mobile sink trajectory, Peer Peer Netw. Appl. 14 (1) (2021) 44–57, http://dx.doi.org/10.1007/s12083-020-00947-w.

[6] A.K. Srivastava, A. Sinha, R. Mishra, S.K. Gupta, EEPMS: Energy efficient path planning for mobile sink in wireless sensor networks: A genetic algorithm-based approach, in: X.-Z. Gao, S. Tiwari, M.C. Trivedi, K.K. Mishra (Eds.), Advances in Computational Intelligence and Communication Technology, Springer Singapore, Singapore, 2021, pp. 101–108.

[7] A. Mehto, S. Tapaswi, K.K. Pattanaik, Optimal rendezvous points selection to reliably acquire data from wireless sensor networks using mobile sink, Computing 103 (4) (2021) 707–733, http://dx.doi.org/10.1007/s00607-021-00917-x.

[8] S.K.S.L. Preetha, R. Dhanalakshmi, P.M. Shakeel, An intelligent approach for energy efficient trajectory design for mobile sink based IoT supported wireless sensor networks, Peer Peer Netw. Appl. 13 (6) (2020) 2011–2022, http://dx.doi.org/10.1007/s12083-019-00798-0.

[9] N. Ghorpade, P. Vijaykarthik, Efficient mobile sink path scheduling for clustered based wireless sensor network, in: IOP Conference Series: Materials Science and Engineering, Vol. 1022, IOP Publishing, 2021, 012084, http://dx.doi.org/10.1088/1757-899x/1022/1/012084.

[10] K. Saxena, N. Gupta, J. Gupta, D.K. Sharma, K. Dev, Trajectory optimization for the UAV assisted data collection in wireless sensor networks, Wirel. Netw. 28 (4) (2022) 1785–1796, http://dx.doi.org/10.1007/s11276-022-02934-w.

[11] P.V.P. Raj, A.M. Khedr, Z.A. Aghbari, EDGO: UAV-based effective data gathering scheme for wireless sensor networks with obstacles, Wirel. Netw. 28 (6) (2022) 2499–2518, http://dx.doi.org/10.1007/s11276-022-02983-1.

[12] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second ed., The MIT Press, 2001, pp. 624–636, URL: http://www.amazon.com/Introduction-Algorithms-Thomas-H-Cormen/dp/0262032937.

[13] J.d.C.V. Rezende, R.I.d. Silva, M.J.F. Souza, Gathering big data in wireless sensor networks by drone, Sensors 20 (23) (2020) http://dx.doi.org/10.3390/s20236954, URL: https://www.mdpi.com/1424-8220/20/23/6954.

[14] R.I. da Silva, M.A. Nascimento, On best drone tour plans for data collection in wireless sensor network, in: Proceedings of the 31st Annual ACM Symposium on Applied Computing, ACM, Pisa, Italy, 2016, pp. 703–708.

[15] M.G.C. Resende, Greedy randomized adaptive search procedures, in: C.A. Floudas, P.M. Pardalos (Eds.), Encyclopedia of Optimization, Second Edition, Springer, 2009, pp. 1460–1469, http://dx.doi.org/10.1007/978-0-387-74759-0_256.

[16] S. Kagi, B.S. Mathapati, Localization in wireless sensor networks: A compact review on state-of-the-art models, in: 2021 6th International Conference on Inventive Computation Technologies (ICICT), 2021, pp. 5–12, http://dx.doi.org/10.1109/ICICT50816.2021.9358793.

[17] S. Sivasakthiselvan, V. Nagarajan, Localization techniques of wireless sensor networks: A review, in: 2020 International Conference on Communication and Signal Processing (ICCSP), 2020, pp. 1643–1648, http://dx.doi.org/10.1109/ICCSP48568.2020.9182290.

[18] R. Bista, J.-W. Chang, Privacy-preserving data aggregation protocols for wireless sensor networks: A survey, Sensors 10 (5) (2010) 4577–4601, http://dx.doi.org/10.3390/s100504577, URL: https://www.mdpi.com/1424-8220/10/5/4577.

[19] R.I. da Silva, C.M. Silva, S. de Oliveira, J.M.S. Nogueira, Using aerial unmanned vehicles for data gathering in wireless sensor networks, in: 2019 IEEE Latin-American Conference on Communications (LATINCOM), 2019, pp. 1–6, http://dx.doi.org/10.1109/LATINCOM48065.2019.8937884.

[20] X. He, X. Fu, Y. Yang, Energy-efficient trajectory planning algorithm based on multi-objective PSO for the mobile sink in wireless sensor networks, IEEE Access 7 (2019) 176204–176217.

[21] T.C. Hung, D. Thi Ngoc, P.T. The, L. Ngoc Hieu, L.N.T. Huynh, L. Dien Tam, A moving direction proposal to save energy consumption for mobile sink in wireless sensor network, in: The 21st International Conference on Advanced Communication Technology (ICACT), IEEE, PyeongChang, South Korea, 2019, pp. 107–110.

[22] S. Yalçin, E. Erdem, A mobile sink path planning for wireless sensor networks based on priority-ordered dependent nonparametric trees, Int. J. Commun. Syst. 33 (12) (2020) http://dx.doi.org/10.1002/dac.4449.

[23] G. Hou, X. Wu, C. Huang, Z. Xu, A new efficient path design algorithm for wireless sensor networks with a mobile sink, in: The 27th Chinese Control and Decision Conference (2015 CCDC), IEEE, 2015, pp. 5972–5977.

[24] J. Chang, J. Jeng, Y. Sheu, Z. Jian, W. Chang, An efficient data collection path planning scheme for wireless sensor networks with mobile sinks, EURASIP J. Wirel. Commun. Netw. 2020 (1) (2020) 257, http://dx.doi.org/10.1186/s13638-020-01873-4.
[25] R. Anwit, A. Tomar, P.K. Jana, Tour planning for multiple mobile sinks in wireless sensor networks: A shark smell optimization approach, Appl. Soft Comput. 97 (Part) (2020) 106802, http://dx.doi.org/10.1016/j.asoc.2020.106802.
[26] J. Chen, J. Tang, UAV-assisted data collection for dynamic and heterogeneous wireless sensor networks, IEEE Wirel. Commun. Lett. 11 (6) (2022) 1288–1292, http://dx.doi.org/10.1109/LWC.2022.3164784.
[27] K. Li, W. Ni, E. Tovar, A. Jamalipour, On-board deep Q-network for UAV-assisted online power transfer and data collection, IEEE Trans. Veh. Technol. 68 (12) (2019) 12215–12226, http://dx.doi.org/10.1109/TVT.2019.2945037.
[28] K. Li, W. Ni, E. Tovar, M. Guizani, Joint flight cruise control and data collection in UAV-aided internet of things: An onboard deep reinforcement learning approach, IEEE Internet Things J. 8 (12) (2021) 9787–9799, http://dx.doi.org/10.1109/JIOT.2020.3019186.
[29] K. Li, W. Ni, F. Dressler, Continuous maneuver control and data capture scheduling of autonomous drone in wireless sensor networks, IEEE Trans. Mob. Comput. (2021) 1, http://dx.doi.org/10.1109/TMC.2021.3049178.
[30] M. López-Ibáñez, J. Dubois-Lacoste, L.P. Cáceres, M. Birattari, T. Stützle, The irace package: Iterated racing for automatic algorithm configuration, Oper. Res. Perspect. 3 (2016) 43–58.
[31] N.M. Razali, Y.B. Wah, Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests, J. Stat. Model. Anal. 2 (1) (2011) 21–33.
[32] H. Hsu, P.A. Lachenbruch, Paired t test, Encycl. Biostat. 6 (2005).

Rone Ilídio da Silva obtained his Bachelor's degree in Computer Science from the Federal University of Ouro Preto (UFOP) in 2002, Brazil, his master's degree in Computer Science from Federal University Fluminense (UFF) in 2004, and doctoral's degree in Computer Science from the Federal University of Minas Gerais (UFMG). He became a professor at the Federal University of Sao Joao del-Rei (UFSJ) in 2009. His research interests are computer networks, wireless sensor networks, ad hoc networks, optimization, metaheuristics, vehicle routing, and augmented reality.

Josiane da Costa Vieira Rezende received her Bachelor's degree in Computer Engineering from the Presidente Antônio Carlos University (UNIPAC) - Brazil, the master's degree in Computer Science from the Federal University of Ouro Preto (UFOP) - Brazil, and Ph.D.'s degree in Computer Science at UFOP - Brasil. She became professor at Fumec University – Minas Gerais in 2020. She has experience in Computer Science, with an emphasis on Optimization and Computational Intelligence. Her main research interests are metaheuristics, combinatorial optimization, and Wireless Sensor Networks.

Marcone Jamilson Freitas Souza obtained his Bachelor's degree in Metallurgical Engineering from the Federal University of Ouro Preto (UFOP), Brazil, and his master's and doctoral degrees in Systems Engineering and Computing from the Federal University of Rio de Janeiro (UFRJ). He became a professor at UFOP in 1982 and retired in December 2018. Since then, he remains active as professor emeritus, being part of the permanent staff of the graduate programs in Computer Science (UFOP), Instrumentation, Control and Automation of Mining Processes (Vale Technological Institute and UFOP), and Mathematical and Computational Modeling of the Federal Center for Technological Education of Minas Gerais (CEFET-MG). He has a research productivity fellowship granted by the Brazilian Council for Scientific and Technological Development (CNPq) in transport and production engineering. His research interests are optimization, metaheuristics, scheduling, timetabling, open-pit mining, vehicle routing, and public transport.