On Best Drone Tour Plans for Data Collection in Wireless Sensor Network*

Rone Ilídio da Silva
Federal Univ. of São João del-Rei, Brazil
rone@ufsj.edu.br

Mario A. Nascimento
University of Alberta, Canada
mario.nascimento@ualberta.ca

ABSTRACT

The benefits of using mobile sinks or data mules for data collection in Wireless Sensor Network (WSN) have been studied in several works. In this paper, we consider a drone with hovering capability as a mobile sink for data collection. Drones, however, have limited power supply that restricts their flying time. Hence, in addition to the WSN's energy cost, the energy cost of the drone itself must be also considered in order to increase the amount of collected data from the WSN. We investigate the problem of determining the best drone tour for data collection in a WSN, i.e., we focus on minimizing the overall drone time, that is the drone flying time *plus* the time needed to collect data from all WSN nodes. We propose two heuristic algorithms to solve this problem. Our experimental results have shown that the proposed heuristics provide performance close to the optimal one in small WSN instances and performance superior to naive strategies in larger instances (where optimal solution is computationally expensive).

CCS Concepts

•**Computer systems organization** → **Embedded systems;** *Redundancy;* Robotics; •**Networks** → Network reliability;

Keywords

wireless sensor network, WSN, mobile sink, path planning, UAV

1. INTRODUCTION

Wireless Sensor Network (WSN) is a computer network composed by typically small nodes having a processor, small memory storage, a set of sensor to collect data, a short range radio for wireless communications and a battery that provides limited autonomy. Usually, the collected data is forwarded through multi-hop routes to a special node, the base station or sink, which typically has no severe limitations such as the other nodes and serves as the interface between the WSN and the user [1]. WSNs using a unique static sink have an early network disconnection time since the nodes located close to the sink deplete their battery quicker than other nodes. This is due to the fact that all data must be forwarded through these nodes to reach the sink. When using a mobile sink, on the other hand, the energy consumption is distributed among other nodes in the WSN, avoiding premature disconnections. Mobile sinks go to the vicinity of different sets of sensor nodes, varying the nodes in its neighborhood. The literature presents several works that consider mobile sinks in order to increase the network lifetime, e.g. [2, 3, 4, 5]. However, they focus on the limitations of the WSN and do not consider limitations of the mobile sink itself.

Unmanned Aerial Vehicles[1] (UAVs), in particular helicopter-like ones, can be used to fly over an WSN and function as a mobile sink for data collection. Nevertheless, UAVs have a limited flying time that must be considered, i.e., using an UAV as a mobile sink within a WSN adds to the complexity of the WSN's operation.

We consider a WSN with the following characteristics: (1) it monitors a square area, (2) inside this area there exists a finite set of 2D points $P = \{p_1, p_2, ..., p_{|P|}\}$ forming a grid that define the *possible drone positions* (PDP), (3) an initial point p_0 where the drone has to start and finish its trip is given and (4) a set of known sensor node $N = \{n_1, n_2, ..., n_{|S|}\}$ is randomly deployed in this area. The nodes in N periodically collect data about this area, storing it in their flash memory. Each node has a radio for data transmission with r meters of range and can form a multi-hop WSN. All sensor nodes also have the same amount of data in their memory (M bytes). We assume a drone as the mobile sink of the WSN which is capable of flying and hovering over the WSN. Finally, assuming the task at hand is to collect all the data from all the WSN nodes, the problem we consider is to find the **Best Drone Tour Plan (BDTP)**, i.e., to find (1) a sequence of

*This research was has been partially supported by CNPq, Brazil and NSERC, Canada.

SAC 2016,April 04-08, 2016, Pisa, Italy

DOI:http://dx.doi.org/10.1145/2851613.2851854

[1]https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle

C collecting points $c_i \in P$ where the drone will hover for collecting data and (2) C disjoint sets of nodes $S_i \subseteq N$, where all nodes in S_i will transmit their data to the drone hovering at the collecting point c_i and $\bigcup_{k=1}^{C} S_k = N$, so that the total time needed to fly from p_0 over the collecting points and return to p_0 (T_{trip}) plus the time to collect all data from the WSN ($T_{collecting}$), which we denote as T_{BDTP}, is minimized.

There are two obvious solutions to this problem: (1) having the drone fly to a single collecting point with all nodes in N sending their data to the drone at that point through a possibly long multi-hop path, or (2) having the drone fly over several (possibly all) collecting points so that all nodes neighboring the collecting points can send their data to the drone via a single-hop data route. The former minimizes T_{trip} at the expense of $T_{collecting}$, whereas the latter does the opposite. Our main goal is to investigate where in between those obvious solutions lies the best (ideally optimal) compromise.

Finding the BDTP is in the class of the NP-complete[2], thus we propose two heuristics to reduce the drone overall time. The first heuristic uses an incremental strategy, in which each interaction adds a new drone position in the result. The second heuristic uses the opposite strategy, from a set of possible drone positions, it eliminates one by one to find the best set. Experiments in small scale showed that incremental strategy presented overall time similar to the optimal solution. Moreover, in larger scenarios, both heuristics presented very similar performance which was superior to naive strategies.

The remainder of this paper is organized as follows. Next, some related works are reviewed. Section 3 describes how to model the problem as graphs and presents three algorithm to solve it. Section 4 presents the simulation results. Finally, Section 5 presents our conclusions and future works.

2. RELATED WORKS

In most of the researches about mobile sink in WSN, the goal is to increase the network lifetime finding the best sink tour. Luo et al. [2] were the first to propose the sink mobility. They analyzed the best mobility pattern and data routing strategy to extend the network lifetime. However, they only considered the sink moving constantly. Wu et al. [6] proposed a cluster-based genetic algorithm to define clusters in order to reduce the sink trip time. Ma et al. [7] considered a set of robots working as data mules for WSNs. Almi'ani et al. [8] assumed that the time between two data collections in a same node cannot be larger than a predefined limit. Notably, all of these researches considered only single-hop data transmissions. Rault et al. [5] and Keskin et al. [4] did consider a multi-hop scenario but focused on the WSN resource only, i.e., they did not consider resource limitations on the mobile sink itself. Finally, Shi et al. [3] focus in a theoretical approach to prove that a time-dependent sink movement problem and flow routing problem can be transformed into a location-dependent problem. However, they did not consider the sink travel time between any two points in their approach. Consequently, the length of the sequence of points forming the sink path on the monitored area did not influence the final result.

[2]This can be proved by a reduction of the Set-covering problem.

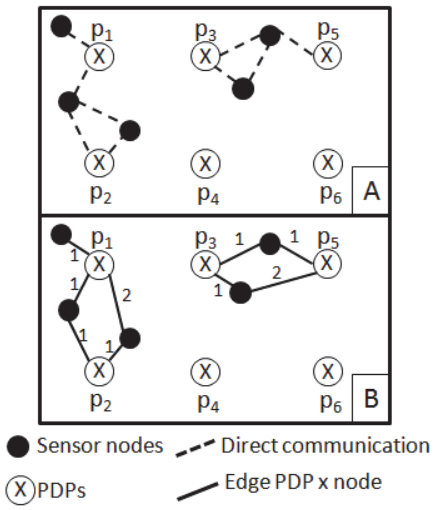

Figure 1: Example of WSN and the correspondent CG.

Unlike these works we consider that the mobile sink (drone) itself has limited resources and therefore its use must be carefully planned, otherwise one risks extending the WSN lifetime but not being able to collect the data it gathered.

3. FINDING THE BDTP

We model the BDTP problem using two graphs. The first graph is called *Trip Grap* (TG) and is used to calculate T_{trip}. We define it as $TG = \{W, F\}$, such that W is a set of vertices representing the PDPs in P and F a set of edges representing the shortest trip between each pair of PDPs. The weights of these edges represent the Euclidean distance between two PDPs. The second graph is called *Connectivity Grap* (CG) and is used to calculate $T_{collecting}$. It is defined as $CG = \{V, E\}$, such that the vertices V represent the sensor nodes and PDPs ($V = P \cup N$). The set of edges E represents possible communication paths between the drone over every PDP in P and all sensor nodes. The weights of these edges represent the distance in hops between each sensor node and the drone over each PDP. There are no edges between vertices representing sensor nodes. Figure 1(A) presents an example of a WSN and a set of PDPs. It also shows the possible direct communications between two sensor nodes and between the drone over each PDP and the sensor nodes. Figure 1(B) shows the corresponding CG graph and the weights of each edge representing the distance in hops between PDPs and sensor nodes.

Having defined those graphs we can now present three algorithms to find the BDTP, one guaranteed to find the optimal solution (but only applicable to small instances of the problem) and two heuristic ones.

3.1 Brute Force Algorithm for The Optimal Solution

The Brute Force algorithm finds the optimal solution for the

BDTP problem. It calculates $T_{collection}$ and T_{trip} for each possible subset R_i of P, in order to find which R_i provides the smallest T_{BDTP}. Algorithm 1 presents the pseudo code. It receives as parameter the sets N and P, the amount of data in each sensor node (M), the network bandwidth (B) and the drone speed (*dspeed*). In every loop iteration, i.e., for every different R_i, the algorithm calculates $T_{collecting}$, T_{trip} , sums them to find T_{BDTP} and verifies if it is the smallest. Finally, it shows the result, which is composed by *bestroute* (the sequence of C PDPs where the drone has to hover over) and *bestst* (a graph where each edge represent a data route from a sensor node to a drone over a specific PDP).

$T_{collecting}$ is calculated in two steps. First, the algorithm calls the function $SpanningTree()$ (line 9) to create st and then it calls the function $CTime()$ (line 10) to calculate $T_{collecting}$. The graph $st \subseteq CG$ is composed by the vertices in R_i, vertices representing all sensor nodes and only one edge connecting each sensor node to a PDP. Every edge in st represents the shortest path (in hops) for each sensor node to send its data to drone hovering over one of the PDPs in R_i. $CTime()$ calculates $T_{collecting}$ adding all weights of the edges in st and multiplying it by the time necessary for one node to transmit all its data over one hop, i.e., $t = \frac{M}{B}$. When st has a vertex representing a sensor node with no edge, $CTime()$ will set $T_{collecting} = \infty$, which means the drone over the PDPs in R_i cannot receive data from all sensor nodes.

To calculate T_{trip}, Algorithm 1 first calls the function $TSP()$ (line 11) to create the best route and then the function $CTime()$(line 12). The function $TSP()$ solves the Traveling Salesperson Problem [9], using R_i. Here, we consider that the initial point $p_0 = (0, 0)$ and the drone has constant speed informed by the user (*dspeed*). The function $CTime()$ calculate T_{trip} dividing the best tour length by *dspeed*.

Algorithm 1 Brute Force Algorithm.

```
1: procedure BRUTE_FORCE(N, P, M, B, dspeed)
2:     besttime ⟵ ∞
3:     bestroute ⟵ ∅
4:     bestst ⟵ ∅
5:     route ⟵ ∅
6:     CG ⟵ CreateCG(N, P)
7:     TG ⟵ CreateTG(P)
8:     for each R_i ⊆ P do
9:         st ⟵ SpanningTree(R_i, CG)
10:        T_colleecting ⟵ CTime(st, M)
11:        route ⟵ TSP(R_i, TG)
12:        T_trip ⟵ CTime(route, TG, dspeed)
13:        T_BDTP ⟵ T_collecting + T_trip
14:        if T_BDTP < besttime then
15:            besttime ⟵ T_BDTP
16:            bestst ⟵ st
17:            bestroute ⟵ route
18:        end if
19:    end for
20:    show(bestroute, bestst)
21: end procedure
```

3.2 Heuristics to BDTP problem

We propose two heuristics based on the metaheuristic GRASP (Greedy Randomized Adaptive Search Procedures) [10]. GRASP defines two phases called **construction** and **local search**. In the former, it creates some possible solutions for the problem using a greedy or a random strategy. In the last phase, the neighborhoods of these solutions are investigated in order to find the local minimum or maximum, depending on the problem. The two proposed heuristics, called **Incremental** and **Decremental**, are different in the construction phase, but have the same local search.

3.2.1 Incremental Heuristic

The Incremental Heuristic creates a set R_i with only one PDP and adds a new PDP each iteration. For every different R_i, the heuristic performs the local search, which calculates T_{BDTP}, and saves the best result. These steps are repeated until the value of T_{BDTP} stop decreasing when R_i increases or $R_i = P$. The heuristic chooses the next PDP (p_j) to add to R_i as the PDP p_j that has the biggest weight and that is at least $d \geq 2 \times r$ distant from the other PDPs in R_i. If there is no PDP with this characteristic, d will be divided by two. Using this minimal distance, the algorithm avoids PDPs close to each other in R_i.

The weight w_i of a PDP p_i is calculated by the equation $w_i = \sum_{h=1}^{z} \frac{hop_h}{h}$, where hop_h is the number of sensor nodes in CG that are connected to p_i by edges with weight equal to h, and z is the biggest weight of the edges that link p_i to sensor nodes in CG. Hence, PDPs on crowded regions receive the highest weights.

Algorithm 2 presents a pseudo code for the algorithm.

Algorithm 2 Construction Phase: Incremental Heuristic

```
1: procedure INCREMENTAL(N, P, M, B, dspeed)
2:     besttime ⟵ ∞
3:     bestroute ⟵ ∅
4:     bestst ⟵ ∅
5:     CG ⟵ CreateCG(N, P)
6:     TG ⟵ CreateTG(P)
7:     w ⟵ CreateWeight(CG, P)
8:     R_i ⟵ ∅
9:     while |R_i| ≤ |P| do
10:        R_i ⟵ R_i ∪ NewPDP(w, R_i, TG)
11:        T_BDTP ⟵ LSearch(R_i, st, route, CG, TG, B)
12:        if T_BDTP < besttime then
13:            besttime ⟵ T_BDTP
14:            bestst ⟵ st
15:            bestroute ⟵ route
16:        else
17:            break
18:        end if
19:    end while
20:    show(bestroute, bestst)
21: end procedure
```

As example, consider the graph CG in Figure 1(B), $r = 60$m and the distance between two PDPs is 84 m. In the first loop iteration, $R_i = \{p_1\}$ since p_1 has the biggest weight $wp_1 = 2.5$. In the second loop iteration, $R_i = \{p_1, p_5\}$, because p_2 and p_3 are less than $d = 2 * r$ (120 m) far from p_1 and $wp_5 = 1.5$. In the third loop interaction, there is no PDP further than $d = 2 \times r$. Hence, $d = d/2$ and $R_i = \{p_1, p_5, p_2\}$, since p_2 has the biggest weight.

3.2.2 Decremental Heuristic

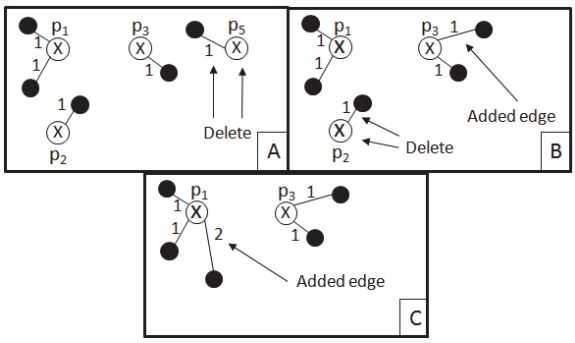

Figure 2: Example of Decremental Heuristic.

The first step of the Decremental heuristic is to create $st \subseteq CG$, a graph with the edges representing the small distance in hops between each sensor node and a PDP. Then, it creates R_i with all PDPs in st. As above the heuristic starts a loop. In each loop iteration, the heuristic removes a PDP from R_i, executes the local search to calculate T_{BDTP} and saves the best result. The loop stops when R_i becomes empty or T_{BDTP} increases when the size of R_i decreases.

The heuristic chooses the PDP to be removed from R_i based on the impact of the removal on the value of $T_{collecting}$. When a PDP p_i is removed from R_i, the edges connecting it to sensor nodes in st also have to be removed. Hence, the nodes connected to p_i must be connected to another PDP. Thus, st receives other edges from CG. The impact of the removal is the difference between the sum of the weights of the new edges and the sum of the weights of the removed edges. Since these new edges have weights equal to or bigger than the removed edges, the value of $T_{collecting}$ keeps the same or increases. However, this removal decreases or keeps T_{trip}, since the drone tour has a smaller number of PDPs to visit. Algorithm 3 presents a pseudo code of this heuristic.

From the graph CG in Figure 1(B), Figure 2-A presents the graph st as example, which shows $R_i = \{p_1, p_2, p_3, p_5\}$. The first PDP to be removed is p_5, which also removes a edge with weight 1. In order to keep the WSN connected, this edge is replaced by another edge with the same weight (Figure 2-B). Hence, the impact of the removal of p_5 is 0. In the second loop iteration, p_2 is removed because it provides impact of removal equal to 1, since the heuristic replaces a edge with weight 1 by another with weight 2 (Figure 2-C). The loop stops in the next iteration because some sensor nodes become unable to send data to a PDP ($T_{BDTP} = \infty$).

3.3 Local Search

The Local Search phase exchanges each PDP in R_i for one of its four neighbor in the grid (up, down, left and right) and returns the smallest T_{BDTP}. It repeats a loop that chooses the first PDP in R_i, replaces this PDP for one of its four neighbor, calculates T_{BDTP} and verifies if it is the smallest. Then, it chooses another neighbor and repeat these operations. At the end, the algorithm returns the smallest T_{BDTP} and, by reference, the sequence of PDPs forming the route (*bestroute*) that provides the smallest T_{BDTP} and the graph *beststs* with edges representing the data routes between each sensor node and the drone over a PDP.

Algorithm 3 Construction Phase: Decremental Heuristic

```
1: procedure DECREMENTAL(N, P, M, B, dspeed)
2:     besttime ⟵ ∞
3:     bestroute ⟵ ⊘
4:     beststs ⟵ ⊘
5:     CG ⟵ CreateCG(N, P)
6:     TG ⟵ CreateTG(P)
7:     st ⟵ SpanningTree(P, CG)
8:     R_i ⟵ ⊘
9:     for each v ∈ P do
10:        if v ∈ st then
11:            R_i ⟵ R_i ∪ v
12:        end if
13:    end for
14:    while |R_i| > 0 do
15:        T_BDTP ⟵ LSearch(R_i, st, route, CG, TG, B)
16:        if T_BDTP < besttime then
17:            besttime ⟵ T_BDTP
18:            beststs ⟵ st
19:            bestroute ⟵ route
20:        else
21:            break
22:        end if
23:    end while
24:    show(bestroute, beststs)
25: end procedure
```

4. SIMULATIONS

We implemented the Force Brute algorithm (Sec. 3.1), the two proposed heuristic (Secs 3.2.1 and 3.2.2) and three other algorithms called OneHop, *Center* and *Position00*, which solve the BDTP problem using naive strategies. The OneHop provides the best solution considering only one-hop transmission. It is essentially the Brute Force using CG without the edges with weights greater than 1. Center assumes only one PDP in the center of the monitored area. Position00 assumes only one PDP in the position 0,0 of the monitoring area (i.e., $T_{trip} = 0$).

The experiments are performed in two phases. In the first phase, we define a small scenario in order to compare the heuristics, the naive algorithms and the optimal solution. In the second phase, we consider a larger monitored area in order to evaluate the performance of these algorithms in WSN with data routes longer than in the first scenarios. In all experiments, the main metric is T_{BDTP}. Every point plotted on the graphs represents the average of 33 simulations using different WSN topologies, which provides 95% of confidence interval.

The mobile sink is a drone able to fly and hover over the WSN. It has a radio with the same range of the nodes (r = 60m). The PDPs form a grid with 84 meters of distance between two of them. Using this grid, every node will be less than 60 meter far from a PDP. For simplicity, we do not consider the time to propagate queries, the drone collects data only when it is hovering, it moves in constant speed, there is only one node transmitting data each time and the network bandwidth is 20 kbps.

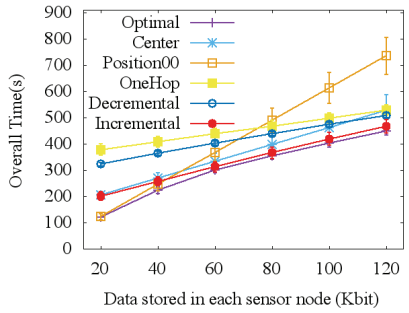

Figure 3: T_{BDTP}

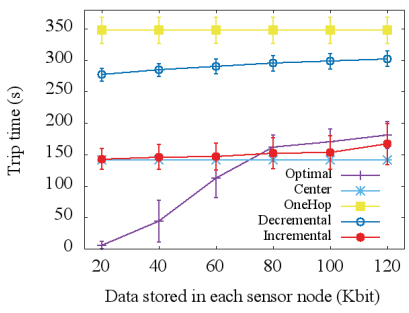

Figure 4: T_{trip}

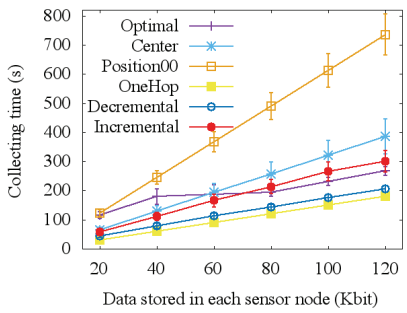

Figure 5: $T_{collecting}$

4.1 Small Monitored Area

We initially consider a 2D square monitored area with 200 meters of side and 30 sensor nodes uniformly randomly deployed. We compare the proposed heuristics with the optimal solution and with the algorithms that use naive strategies. We increase the amount of data storage in each sensor node and plot these values in the X axis. In order to analyze the behavior of T_{BDTP}, we present three graphs. The first shows T_{BDTP} in its Y axis (Figure 3), the second shows T_{trip} (Figure 4) and the last $T_{collecting}$ (Figure 5). The performance of the algorithms in the first graph is analyzed with the information provided by the other two graphs.

In the first graph (Figure 3) we verify that T_{BDTP} provided by all algorithms increase when the amount of data in the sensor nodes increases, as expected. The optimal solution behavior shows that the algorithms must increase the number of PDPs in the drone tour and, consequently, increase T_{trip} to avoid the quick growing of $T_{collecting}$ and keep T_{BDTP} as low as possible. When the amount of data is low, the time necessary to transmit it to the sink is short. Hence, the best strategy is to keep the drone in the initial point where $T_{trip} = 0$, such as Position00. When the amount of data increases, the best strategy is to move the drone in order to reduce the number of hops in the data routes. This increases T_{trip}, but avoids $T_{collecting}$ increasing quickly.

OneHop provides high T_{BDTP} because its $T_{collecting}$ is high. Since this algorithm consider only one-hop transmissions, the drone tour have to be composed by several PDPs. This characteristics provides the smallest possible $T_{collecting}$, but increases T_{trip}. Position00 increases T_{BDTP} quickly when the amount of the data increases, since the data must be transmitted in multi-hop routes. The result of this algorithm is not plotted in Figure 4 because it always provides $T_{trip} = 0$. Center algorithm provides average performance if compared with the other algorithms. This is due the small monitored area. The drone always does a short trip (141.4 m) and the majority of the sensor nodes can send their data to drone over only one-hop or two-hop data routes.

Decremental heuristic does not present good performance in small scenarios, varying T_{BDTP} from 14% to 265% of the optimal. Its strategy tends to provide drone tours with more PDPs than the Incremental heuristic and, consequently, larger values of T_{trip}. It can find reasonable solutions only when the sensor nodes have large amount of data to transmit. Incremental provides performance close to the optimal solution in almost all experiments, varying from 4% (large amount of data stored in the sensor nodes) to 65% (small amount of data stored in the sensor nodes) of the optimal solution. Its strategy of decrease the number of PDPs in the solution provides small sets of PDPS and, consequently, small values of T_{trip}. These two heuristics slightly increase T_{trip} to reduce the quickly growth of $T_{collection}$.

4.2 Larger Monitored Area

We now consider a 2D square monitored area with 400 meters of side. We do not consider Brute Force and OneHop algorithms because in larger settings they become too expensive to be practical. In Figure 6 we increase the amount of data in each sensor node (X axis) and analyze T_{BDTP} (Y axis). Center and Position00 increase T_{BDTP} two or three times faster than the heuristics, respectively, due to their long data routes. Decremental heuristic presents better performance here than in the first experiments and sometimes better than the Incremental heuristic. This is due to the fact that its strategy tends to create drone tours with more PDPs than the Incremental. This provides good results in larger monitored area, since more PDPs in the drone tours decrease the number of hops in the data routes.

Figure 7 presents the result of the experiments in which we increase the number of sensor nodes from 100 to 250 (X axis) and show T_{BDTP} (Y axis). We consider 60 Kbit of data in each sensor node and the drone speed is 2 m/s. The naive algorithms do not have good performance (from 40% to 290% worse than the heuristics) because they create long data routes to deliver data to sink. The two proposed heuristics have better performance because they increased the drone tour to reduce the growth of $T_{collecting}$.

Figure 8 presents experiments to analyze the influence of the drone's speed. We varied it from 0.5 m/s to 3 m/s and plotted in the X axis. We consider 60 Kbit of data in each sensor node and the number of nodes in the monitored area is 150. Position00 do not change the sink position, hence T_{BDTP} does not change. Center improved its performance because the drone has to move to the center of the monitored area. The proposed heuristics take advantage of the bigger drone speed. They increase the number of PDPS in the drone tour to reduce $T_{collection}$. The two heuristics have similar behavior, except when the drone speed is small, because Decremental tends to provide tours with more PDPs and the drone must be fast to keep T_{trip} low.

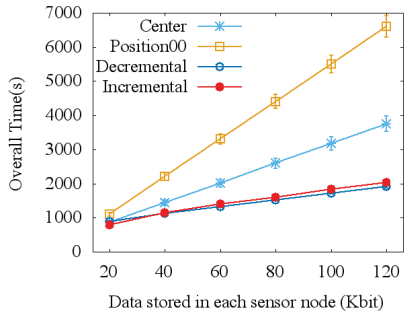

Figure 6: Increasing the storage data.

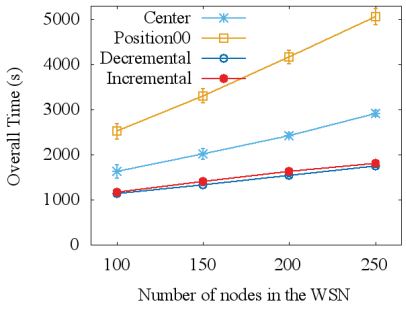

Figure 7: Increasing number of nodes.

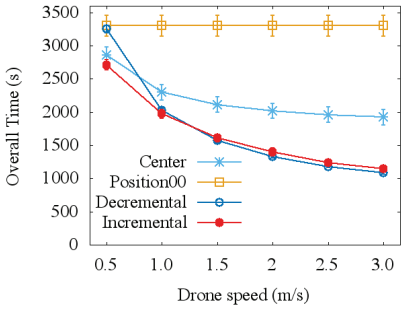

Figure 8: Increasing the drone speed.

5. CONCLUSIONS AND FUTURE WORKS

This work analyzed the problem of finding the Best Drone Tour Plan for data collecting in WSN. We considered a quadcopter-like UAV as the mobile sink, which is able collect data from all nodes of a disconnected WSN, even in dangerous or inaccessible regions. However, it has a flying time limited by its battery. We focused in to find the tour plan that provides the smallest time for the drone to collect data from all sensor nodes. In this way, we faced a tradeoff between the drone trip time and collecting time in order to find the best overall time.

We defined the Brute Force algorithm to provide the optimal solution, two heuristics called Incremental and Decremental and three algorithms (OneHop, Center and Position00) that use naive strategies. We created simulations to analyze the performance of the algorithms. Center and Position00 provided small T_{BDTP} only when the amount of data stored in the sensor nodes was small. When the amount of data increased, $T_{collecting}$ increased quickly because this algorithms create long data routes to delivery data to sink. The OneHop could not find good solutions because it created long drone tours, which increased T_{trip}. The Incremental heuristic had performance close to the optimal solutions in small monitored areas. In larger monitored area, the Incremental and Decremental had similar performance. They were better than the naive algorithms in practically all experiments because they found a balance between T_{trip} and $T_{collecting}$.

Since the value of $T_{collecting}$ is directly proportional to the number of hops in the data route transmissions, we believe that when $T_{collecting}$ is reduced, the energy consumption in the WSN is reduced too. The radios in the sensor nodes are responsible by the consumption of the largest amount of the energy in WSN. The reduction of the data route lengths decreases both the time needed to transmit data to sink and the number of transmissions, saving energy. However, this analysis will be made in future work. We also intend to define a protocol to disseminate queries into the WSN and to forward data onto sink. This protocol and the heuristics to define the drone tour must work together in order to reduce the overall time and save energy in the WSN.

6. REFERENCES

[1] F. Zhao and L. J. Guibas, *Wireless sensor networks : an information processing approach*, ser. The Morgan Kaufmann series in networking., Hardcover, Ed. Morgan Kaufmann, 2004.

[2] J.-P. Jun Luo ;Hubaux, "Joint mobility and routing for lifetime elongation in wireless sensor networks," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 3. IEEE, March 2005, pp. 1735 – 1746.

[3] Y. Yi Shi; Hou, "Some fundamental results on base station movement problem for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 4, pp. 1054 – 1067, August 2012.

[4] N. A. M. Emre Keskin, I. Kuban Altinel and C. Ersoy, "Wireless sensor network lifetime maximization by optimal sensor deployment, activity scheduling, data routing and sink mobility," *Ad Hoc Networks*, vol. 17, pp. 18–36, 2014.

[5] T. Rault, A. Bouabdallah, and Y. Challal, "WSN Lifetime Optimization through Controlled Sink Mobility and Packet Bufferization," in *The 4th Global Information Infrastructure and Networking Symposium*, Trento, Italy, 2013.

[6] J. L. SY Wu, "Evolutionary path planning of a data mule in wireless sensor network by using shortcuts," in *IEEE Congress on Evolutionary Computation*. IEEE, July 2014.

[7] M. Ma, Y. Yang, and M. Zhao, "Tour planning for mobile data-gathering mechanisms in wireless sensor networks." *IEEE T. Vehicular Technology*, vol. 62, no. 4, pp. 1472–1483, May-2013 2013.

[8] K. Almiani, A. Viglas, and L. Libman, "Mobile element path planning for time-constrained data gathering in wireless sensor networks." in *AINA*. IEEE Computer Society, 2010, pp. 843–850.

[9] R. L. R. Thomas H. Cormen, Clifford Stein and C. E. Leiserson, *Introduction to Algorithms*, 3rd ed., McGraw-Hill, Ed. McGraw-Hill Higher Education, 2009.

[10] T. A. Feo and M. G. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, pp. 109–133, 1995.