Noname manuscript No.
(will be inserted by the editor)

Locality sensitive heuristics for solving the Data Mule Routing Problem

Pablo Luiz Araújo Munhoz
Uverton dos Santos Souza
Pedro Henrique González
Luiz Satoru Ochi
Philippe Michelon
Lcia Maria de A. Drummond

Received: date / Accepted: date

Abstract A usual way to collect data in a Wireless Sensor Network (WSN) is by the support of a special agent, called data mule, that moves between sensor nodes and performs all communication between them. In this work, the focus is on the construction of the route that the data mule must follow to serve all nodes in the WSN. This paper deals with the case when the data mule does not have a global view of the network, i.e., a prior knowledge of the network as a whole. Thus, at each node, the data mule makes a decision about the next node to be visited based only on a limited local knowledge of the WSN. Considering this realist scenario, two locality sensitive heuristics are proposed. These heuristics differ by the criterion of choice of the next visited node, while the first one uses a simpler greedy choice, the second one uses the geometric concept of convex hull. They were executed in instances of the literature and their results were compared both in terms of route length and in number of sent messages as well. Some theoretical results, a mathematical formulation, and some lower bounds for the global view scenario are also proposed, in order to provide some parameters to evaluate the quality of the solutions given by the proposed heuristics. The obtained results show that the proposed heuristics give good solutions in a reasonable time when compared with the optimal solutions and lower bounds.

Munhoz, P.L.A; Souza, U.S.; Ochi,L.S; Drummond, L.M.A
Universidade Federal Fluminense
Instituto de Computação
E-mail: {pmunhoz,ueverton,satoru,lucia}@ic.uff.br

Gonzlez, P.H
Universidade Federal do Rio de Janeiro
Programa de Engenharia de Sistemas e Computação
E-mail: pegonzalez@cos.ufrj.br

Michelon, P.
Université d'Avignon et de Pays de Vaucluse
Laboratoire Informatique dAvignon
E-mail: philippe.michelon@univ-avignon.fr

Keywords Data Mule · Routing Problem · Locality Sensitive · Heuristics · Convex hull

1 Introduction

Wireless Sensor Networks, WSN, have received much attention in last decades due to its flexibility and its easy and fast deployment capability. In addition, there are many practical applications where WSN can be used, as, for example, environmental monitoring and military applications [15,23]. In this kind of network, communication is exclusively wireless and information exchange is accomplished when there is intersection between sensors spatial coverages. The information routing is one of the main problems of WSN [21,24]. In some cases, a mobile agent, called date mule, is responsible for performing the network communication. Data mule is a mobile agent that has greater processing, memory capacities and energy availability than regular sensors of the WSN. It is responsible for collecting data from all sensors and take them to a base station, reducing the number of exchanged messages in the network and, consequently, the spent energy for data transmission.

This work considers that sensors are distributed in a bi-dimensional space and have communication range equal to r. In that scenario, the data mule has to serve each node of the WSN, by sending or receiving data to/from each of them. The data mule has no knowledge about the global network and should visit a minimum number of nodes necessary to serve all nodes demands. At each node, it has to decide the next one to be visited, aiming at the minimization of the total route. The goal of using a data mule and minimizing its route is to minimize the energy consumption of the WSN. Note that the visited nodes form a connected dominant set. Remark also that at each node, the data mule covers only a limited set of other sensor nodes, and having that limited knowledge about the nodes, it has to choose the best one to be visited next.

Thus, our problem follows the next basic assumptions. Let $G = (V, E)$ be a graph representing a network and the geographical position of a node is given by euclidean coordinates, i.e., $V(G)$ is a set of points placed in an Euclidean plan, and each edge $(i, j) \in E(G)$ between two vertexes i and j exists if the corresponding sensors, that they represent, are within their communication range. The set $N(i)$ contains the neighbour nodes of vertex i, and the corresponding sensor only knows the other sensors in the neighbourhood and their corresponding euclidean coordinates. Edges have no weights. Let $s \in V$ be a vertex, from where the data mule initiates and finishes the route, called here, base station. The data mule moves between nodes, and only serves a node i when located in some node $j \in N(i)$. The data mule can serve every node that is in the neighborhood, and not only the node where it is. When a mule moves to a node from another one, the corresponding edge is included in the route. An edge can be used by the data mule more than once. Each time the edge is used, it is included in the route.

The objective of the problem treated in this work is minimizing the route length traversed by the data mule, that visits a subset of nodes $D \subseteq V$ forming a connected dominant set of G. The remaining of this work is organized as follows.

Related works are presented in Section 2. Section 3 presents some theoretical remarks on Data Mule Routing problem (DMRP), we show that when the input is allowed to be a general graph even the mule having a global view of the network, DMRP cannot be approximated in polynomial time to within a factor of $(1 - o(1)) \log n$ (unless P=NP). However, we remark that the set of realistic instances treated in this work, in fact, coincides with the Unit Disk Graph class, and then could be approximated in polynomial time to within a constant factor if the mule has a global view, suggesting that the geometric structure of Unit Disk Graphs could be explored to solve our realistic scenario where the mule has only local view. A mathematical formulation for the global view case is also presented in this section. In Section 4, we present two heuristics to work on the the realistic scenario where the data mule has only a local view of the network. The heuristics differ from each other in the criterion adopted to choose each next node in the route. The first algorithm adopts a simpler greedy approach, while the second one is based on a geometrical view of the convex-hull of the sub-graph formed by the nodes covered by the data mule at each node. Results and analyses are shown in Section 5. At last, in Section 6, concluding remarks are presented.

2 Related Work

The communication in WSNs can be performed basically either by using virtual backbones and its own network infra-structure, or by using a mobile agent, called data mule. The problem of constructing virtual backbones for communication can be modeled as a Minimum Connected Dominating Set Problem, MCDS, as can be seen in [1,8,14,25].

Papers from related literature of WSN that aim at solving the MCDS, usually present distributed algorithms and consider simultaneous communication among sensor nodes, and local processing at each node. When the communication is performed through a data mule, most works focus on the Data Mule Routing Problem, an $\mathcal{NP}$-hard problem [33]. Usually, those related papers consider that the data mule has the complete knowledge of the WSN. Considering that, mathematical formulations and approximate heuristics were proposed to solve the problem [2,27,34].

Thus, this section is divided in two subsections, one that describes the works that consider the problem as a MCDS, and the other that presents papers related only to the Data Mule Routing Problem.

2.1 Connected Dominating Set based Algorithms

In this section, related works, in which the data transmission on the network is modeled as the dominant set problem, are presented. When this approach is considered, network nodes themselves are able to transmit their data to the base station, constituting, consequently, a virtual backbone of communication.

In [6] two distributed algorithms, based on a sequential algorithm for solving the Connected Dominating Set Problem (CDSP) [13], applied in undirected graphs are presented. The proposed algorithms consider possibles changes in the networks and are re-executed periodically to keep the connectivity of the current network. They consider that each node has complete knowledge of the network, i.e., it has a copy of the entire graph that represents the network. Those algorithms also employ distributed spanning tree algorithms as building blocks to solve the problem and use shortest path algorithms between all pair of nodes.

In the distributed algorithm proposed in [29], to solve the CDSP, each node broadcasts its set of neighbours. A neighborhood is defined by using a disk unit graph representation. A node defines itself as a dominant, if it has two non-adjacent neighbours. Then, another broadcast round is executed so that the dominant nodes are known by all other nodes. This step occurs using a 2-hop communication between all nodes. After that, there is another step to eliminate redundancies. The algorithm is tested in synthetic instances and its results are compared with the others from the related literature. Moreover, some variations, considering topology changes, like node inclusions, eliminations and movements, are also introduced.

In [1] a distributed algorithm with two phases using a disk unit graph representation was proposed. In the first phase, a maximal independent set (MIS) is created. Although MIS is a dominant set, it is not connected yet. So, in the second phase, a spanning tree algorithm is executed over the MIS, aiming not only to connect those nodes but also to remove eventual extra nodes. The spanning tree algorithm also includes a node defined as base station, even when it does not belong to MIS and only 1-hop communication is allowed.

In [8], the goal of the problem is defining time slots when each node can communicate with the data mule without communication conflicts, i.e, two nodes communicating with the data mule simultaneously. This work uses the coloring approach proposed in [9] to define the connected set and an independent set. It also analyses the relation between the sizes of MIS and Minimum Connected Dominant Set, contributing to define best bounds to several similar problems.

In [14] the CDSP is employed to define virtual backbones for communication in a WSN. For this purpose a 1-hop distributed two-phase algorithm was developed and it is applied in a network represented as a unit disk graph in a Cartesian system. This algorithm uses a color scheme, computation of the convex hull of the nodes and MIS to define a CDS for the network.

In [11] a distributed algorithm is developed for the CDSP where each node of the network has a weight and communication congestions can occur in the

network. The objective of the problem is building a CDS with the lowest possible weight. In order to solve this problem, the author uses a 1-hop CONGEST model [20] and proves that the MCDS can be solved by using it.

In [32] a mobile agent is responsible for collecting data from a WSN. In addition to data collection, the cost of transmission according to the capacity of communication channel and the time spent at each point to collect data are taken into account. In this problem, an undirected graph represents the network and some *archor* points must be visited by the data mule in order to attend all nodes' demands. The authors divide the problem in two stages, the first one where the mule decides the *archor* points that it will visit not exceeding a bounded time, and the second one, where, given the mule route, the sensors decide the amount of data that they will transmit to the mule in each *archor* point chosen by the mule in the first stage. For both decisions, mathematical formulations with a global view of the network were proposed.

Table 1 summarizes those related works, presenting their main aspects.

2.2 Data Mule based Algorithms

In this section a literature review is presented for the case where a Data Mule is responsible for all communication in the network.

In [33] the Message Ferrying is defined as the agent responsible for collecting and transmiting the information between the other sensors. In this paper, it is proved that the problem of routing the Message Ferrying (Data Mule) is $\mathcal{NP}$-hard through a generalization of the Traveling Salesman Problem (TSP [10]). The objective is reducing the average delay in message deliveries. An algorithm has been developed for the TSP with a refinement phase that applies movements to reduce the delay messages exchanges.

In [34] the problem defined in [33] is also tackled. But, here, it uses multiple Message Ferries that can communicate with each other. The communication redundancy provided by the multiple Message Ferries introduces a fault tolerance mechanism in that scenario. Tests were executed considering different types of communication between the ferries (restricted or free), and a unique route or different routes.

A problem using a data mule that can choose between various speeds and where the sensors can move was treated in [2]. Besides those previously described features, the difficulty of the problem has increased because the network sensors move arbitrarily within a predetermined area, and it is necessary to define the speed of the data mule which will allow for it to serve all sensors. Because it is not possible to guarantee that there will always be contact between the data mule and the sensors, an algorithm, that uses the model of movement of the sensors to define a contact probability of the data mule with the sensors, was proposed.

In [31], a robot that collects sensor data and returns to the base station is used. The communication range of the antennas of each sensor is determined by the amount of battery that each sensor has, and reduces with the use

of the battery. The problem is a generalization of the Traveling Salesman Problem with Neighborhoods ([12]), where customers are willing to meet with the traveling salesman within a certain distance of their residence. A two-phase evolutionary algorithm was developed. The first phase solves the TSP, using the position of the sensors, and the second phase defines the best meeting points within the communication range of each sensor.

In [16], a problem with only one data mule, named SenCar, and static sensors is addressed. This problem considers multi-hop communication, i.e., sensors can transmit their messages through other sensors. A heuristic algorithm based on clustering to balance the distance traveled by SenCar and the traffic load was proposed, in order to increase the network lifetime. The algorithm works both in connected and disconnected networks and it was evaluated by simulations.

In [30], a problem with multiple data mules and multi-hop communication is presented. Concentrator sensors, called *rendezvous points* (RP), are defined and only they must be served by the data mule. Thus, the mule does not have to travel long distances to collect the data of all sensors. The authors developed two heuristics to address this problem. In the first algorithm the mules choose the RP when they are moving in the routing tree, while in the second algorithm RP are chosen by considering the maximum energy saving and travel distance ratios.

In [22] there is no prior knowledge of the location of the sensors which must be served, so the data mule executes a location collection procedure to establish which sensors will be used to define the service route. There is also the possibility of communication between the sensors using a k-hop communication, where k defines the highest degree of communication between sensors, i.e., how many sensors a message can pass before reaching the data mule. The solutions are evaluated considering the delay for data transmission and energy efficiency. A heuristic based on the Ant Colony meta-heuristic was developed and tests were carried out in a network simulator.

In [19], multiple data mules were used, each one moving in an independent route at constant or variable speeds. In this work, the sensors are static, but each of them receives a weight that defines their priority to be served. The weights are collected by the data mules and can be updated at each service according the necessity to visit the sensor. A sequential and distributed versions of a probabilistic algorithm, based on a TSP approach, were developed.

In [26] and [27], the Data Mule Scheduling Problem is presented. In this problem, two more perspectives are added to the basic problem of routing: speed control and the order of sensors serving. They developed a mathematical model to treat perspective separately, and heuristics that solve the problems independently.

In [28], sensor nodes send the information to sinks where the data mule, here called mobile sink, must pass to collect the information from the network. There is a motion restriction because the mobile sink is viewed as an unmanned aerial vehicle (UAV), thus the route should be smooth, according to the turning constraints that the UAV is capable of doing. A constructive heuristics

was proposed to deal with this problem and the results are compared with a classical TSP solution and a traditional in-network communication routing.

In [17], the data mule, called an M-collector, must start and return to the data sink (base station) collecting all data from the network. So, pooling points where data is transmitted from sensor nodes to the data mule are defined. The problem consists in defining the best pooling points and the best route between them. The candidate pooling points can be defined either by the network nodes themselves or by a previous stage where one or more M-collectors explore the network by searching for points where there is communication with some node of the network. The final path of the M-collector is formed by the set of pooling points that it passes to serve all the nodes of the network. A mathematical formulation and a greedy two-phase heuristic are proposed. The case where multiple M-collectors can be used was also treated.

A summary of the related work that uses a data mule is presented in Table 2.

In our work, data are collected from static sensors by a single data mule that starts and finishes its route in a base station, using end-to-end communication. Remark that we consider that the mule's route is a closed walk instead of a hamiltonian cycle (as considered in all related works), i.e, given a route, the mule can visit each vertex more than once.

Table 1 Connected Dominant Set based algorithms

Author	Graph Rep.	Neig. Knowledge	Algorithm
Das and Bharghavan (1997) [6]	Undirected	Global	All pairs shortest path
Wu and Li (1999) [29]	Unit Disk	2-hop	All pairs shortest path for a distributed DS calculation
Alzoubi *et al.* (2002) [1]	Unit Disk	1-hop	MIS and dominating tree
Funke *et al.* (2006) [8]	Unit Disk	1-hop	CDS using distance-2-coloring algorithm
Islam *et al.* (2008) [14]	Unit Disk	1-hop	CDS using convex-hull and MIS
Ghaffari (2014) [11]	Weigth Undirected	1-hop	CONGEST Model based in DS
Zhao *et al.* (2015) [32]	Undirected	Global	Distributed decomposition using Mathematical Formulations

Table 2 Data mule based algorithms

Authors	Sensors Movement	Data Mule	Characteristics		Algorithm
			Dest.	Communication	
Zhao and Ammar (2003)[33]	Static	Single	Sensors	End-to-end	Exact algorithm
Zhao *et al.* (2005)[34]	Static	Multiple	Sensors	End-to-end	Assign and route algorithm for TSP
Bin Tariq *et al.* (2006)[2]	Dynamic	Single	Sensors	End-to-end	Optimized Waypoints
Yuan *et al.* 2007[31]	Static	Single	Base Station	End-to-end	Evolutionary Algorithm
Ma and Yang (2007)[16]	Static	Single	Base Station	End-to-end	Clustering Algorithm
Xing *et al.* (2008)[30]	Static	Single	Sinks	Multi-hop	Routing tree with pickup points
Rao *et al.* (2008)[22]	Dynamic	Single	Base Station	Multi-hop	Ant colony
Ngai *et al.* (2009)[19]	Static	Multiple	Base Station	End-to-end	Spanning tree based Algorithm
Sugihara *et al.* (2010)[26]	Static	Single	Base Station	End-to-end	Mathematical Formulation
Sugihara *et al.* (2011)[27]	Static	Single	Base Station	End-to-end	Shortest Path
Wichmann (2012) [28]	Static	Single	Sinks	End-to-end	Constructive heuristics
Ma *et al.* (2013)[17]	Static	Multiple	Base Station	End-to-end	Spanning tree covering algorithm

3 Theoretical Remarks

In order to demonstrate how difficult it is to find a route for a data mule, we will first perform an analysis of the problem when the data mule has a global view of the network, that is, the data mule knows the underlying graph G representing the network. In such a scenario, the data mule's aim is to solve the following problem.

	Data Mule with Global View
Input:	A graph G, and a base station node $v \in V(G)$.
Goal:	Determine a minimum closed walk W of G such that $v \in V(W)$, and for all node $x \in V(G)$, $N[x] \cap V(W) \neq \emptyset$. That is, either $x \in V(W)$ or some neighbor y of x belongs to $V(W)$.

Next result shows that if it is assumed that G is a general graph then it is very unlikely to exist an algorithm to find in polynomial time routes for a data mule with cost near to the optimal solution.

Theorem 1 Data Mule with Global View *on general graphs cannot be approximated in polynomial time to within a factor of* $(1 - o(1)).\ln n$*, where* $n = |V(G)|$*, unless* $NP \subseteq DTIME(n^{O(\log\log n)})$.

Proof Given a set of elements $U = 1, 2, \ldots, n$ (universe) and a collection S of m sets whose union equals the universe, the Set Cover problem is to identify the smallest sub-collection of S whose union equals the universe. Set Cover cannot be approximated in polynomial time to within a factor of $(1 - o(1)).\ln|U|$, unless $NP \subseteq DTIME(n^{O(\log\log n)})$ [7].

Now, we present a L-reduction from Set Cover to Data Mule with Global View. Let $V(G) = \{v_u : u \in U\} \cup \{v_s : S \in S\} \cup \{v\}$. If $u \in S$ then add edge (v_u, v_s) in $E(G)$. Finally add edges in $E(G)$ in order to obtain a clique induced by $\{v_{s'} : S' \in S\} \cup \{v\}$. Figure 1 illustrates a graph G constructed from an instance of Set Cover.

If C has a sub-collection C' of size k then $\{v\} \cup \{v_s : S \in C\}$ induces a clique of G of size $k + 1$, and then contains a cycle of length $k + 1$ such that any vertex $w \notin \{v\} \cup \{v_s : S \in C\}$ has at least one neighbor in such a cycle. Conversely, if G has such a closed walk Q of order $k + 1$, without loss of generality Q has no vertex of $\{v_u : u \in U\}$, then $V(Q) \setminus \{v\}$ represent a sub-collection of C of size at most k which is a set cover of U.

To transfer the approximation lower bound of Set Cover to Data Mule with Global View we need such hardness result on instances of set cover satisfying $ln(|U| + |S|) \approx ln(|U|)$. However, as first observed in [4], it turns out that this is indeed true analyzing of Feiges construction [7]. ■

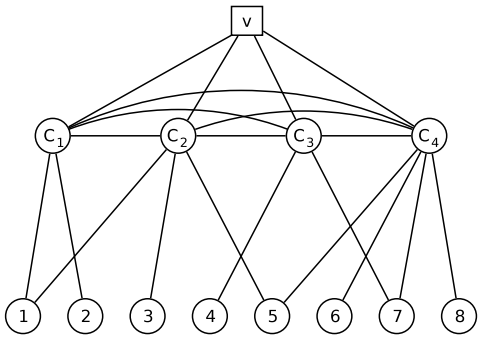

Fig. 1 Graph G obtained from $U = \{1, 2, \ldots, 8\}$ and $S = \{1, 2\}, \{1, 3, 5\}, \{4, 7\}, \{5, 6, 7, 8\}$.

The above result illustrates the hardness of approximate data mule problems on general graphs. Next result shows that the realistic instances of our problem belongs to a very special graph class.

Definition 1 A *unit disk* graph is the intersection graph of a family of disks of unit size in the Euclidean plane.

Lemma 1 *Any realistic instance G of* Data Mule with Global View *is a unit disk graph.*

Proof Definition 1 is the classical definition of unit disk graph. However, through a isotropic scaling in intersection model it is easy to see that Unit Disk Graphs coincide with the intersection graphs of a family of disks of *same* size in the Euclidean plane.

The instances of our problem are graphs G representing a network node (sensor with the same reachability) whose geographical position is given by euclidean coordinates, i.e., $V(G)$ is a set of points placed in an Euclidean plan, and each edge $(i, j) \in E(G)$ between two vertices i and j exists if the corresponding sensors, that they represent, are within their communication range of each other.

A sensor's communication range can be seen as a disk of radius equal to a constant r, thus data mule instances are composed by graphs with one vertex for each disk, and with an edge between two vertices whenever the corresponding vertices lie within a distance at most r of each other. Such graphs coincide with the intersection graphs of a family of disks of radius equals $\frac{r}{2}$. ■

The next result illustrates an interesting property of our problem in practical instances.

Lemma 2 Data Mule with Global View *on Unit Disk Graphs admits a $(2 + \epsilon)$-approximation algorithm, $\epsilon > 0$.*

Proof Any data mule route contains a connected dominating set. On the other hand, from a connected dominating set S it is easy to obtain a data mule route

of size $2|S|-2$. As Connected Dominating Set $(1+\epsilon)$-approximation algorithm [3], then using such algorithm as subroutine we obtain an approximated solution for Data Mule with Global View to within a factor of $(2+\epsilon)$. ■

3.1 Lower Bounds

Now, we are interested to recognize some lower bounds for a solution of Data Mule with Global View that can be found by efficient algorithms.

Given G be a simple graph, and a base station $v \in V(G)$. We denote $d(v,w)$ as the distance between v and w in G, and $d_v = \max_{w \in V(G)} d(v,w)$.

Lemma 3 *Let $OPT(G,v)$ be an optimal solution value for* Data Mule with Global View *on G with base station v. It holds that $OPT(G,v) \geq 2(d_v - 1)$.*

And such lower bound can be found in $O(m)$ time.

Proof Let w be a vertex of $V(G)$ such that $d_v = d(v,w)$ (w and d_v can be found by a breadth-first search). Clearly the data mule needs to travel through at least $d_v - 1$ edges in order to attend w. After that, the data mule will travel through some edges, and will return to base station which spend at least more $d_v - 1$ steps. ■

Now we present a hierarchy of lower bounds for the problem, whose values and performance to be found depends of an input integer k.

Lemma 4 *Given G, v and an integer $k \geq 1$. Let S be a set composed by the k most distant vertices of v, and T_k be a steiner tree to connect $\{v\} \cup S$. Let $LB_k = |T_k| - k + \min_{w \in S} d(v,w) - 1$. For all $k \geq 1$, it holds that*

$$OPT(G,v) \geq LB_k.$$

Proof Note that $LB_1 = 2(d_v - 1)$. As previously, the data mule needs to travel through some edges in order to attend all vertices in S, which walks at least $|T_K| - k$ edges. As the data mule must return to base station then at least more $\min_{w \in S} d(v,w) - 1$ steps must be done. ■

To compute LB_k, $k > 1$, we have to solve a Steiner instance as subroutine. As Steiner Tree is NP-hard then such strategy is viable only for very small values of k. However even $k = 2$ is already able to improve our previous lower bound. In special, LB_2 can also be quickly found as described below.

Lemma 5 *$LB_{k=2}$ can be computed in $O(m.n)$ time.*

Proof As $k = 2$ then the Steiner subroutine must connect only three terminals. Thus, at most one vertex, say u, of such Steiner tree has degree greater than two (equal to 3). Hence finding all shortest path between all pair of vertices, which can be done in $O(n.m)$ time, such Steiner tree can be constructed. Note that, given u, if any, the path from a terminal to u is a shortest path, and there exists only $O(n)$ possibilities for u. ■

3.2 Mathematical Formulation

A mathematical formulation, that considers all the characteristics of the problem is proposed in this subsection. Although the use of mathematical formulation is not the most appropriate approach for the Data Mule Routing Problem, it is a key technique for obtaining bounds. It is important to notice that in the problem solved using the formulation all information about the network is available, allowing to obtain a better path and consequently a reduction in the number of mule movements used to serve all sensor nodes.

Let $G'(V, A)$ be a bidirected graph formed by bidirectiong the edges of original graph $G(V, E)$. The set of vertices $V(G)$ are maintained as points in an Euclidean plane as the original graph. The set of arcs $(i, j) \in A$, represents the possible paths that the mule can choose. Two sets of variables are defined, x_{ij} and y_i. The variables x_{ij} are 1 if the data mule uses the arc (i, j) in his path, and 0 otherwise. The other binary variables y_i are set to 1 if the vertex $i \in V$ are visited by the mule, and 0 if it is attended by another node.

$$\min \quad \sum_{(i,j)\in A} x_{ij} \tag{1}$$

$$\text{s.t.} \quad \sum_{j\in N(i)\cup\{i\}} y_j \geq 1, \forall i \in V \tag{2}$$

$$\sum_{j\in\delta^+(i)} x_{ij} \geq y_i, \forall i \in V \tag{3}$$

$$\sum_{j\in\delta^-(i)} x_{ji} \geq y_i, \forall i \in V \tag{4}$$

$$\sum_{j\in\delta^+(i)} x_{ij} = \sum_{j\in\delta^-(i)} x_{ji}, \forall i \in V \tag{5}$$

$$\sum_{j\in\delta^+(i)} x_{ij} \leq |N(i)|y_i, \forall i \in V \tag{6}$$

$$\sum_{j\in\delta^-(i)} x_{ji} \leq |N(i)|y_i, \forall i \in V \tag{7}$$

$$y_0 = 1 \tag{8}$$

$$\sum_{i\in\bar{S}}\sum_{j\in S} x_{ij} \geq y_s, \forall S \subseteq V \setminus \{0\}, s \in S \tag{9}$$

$$x_{ij} \in \{0, 1\}, \forall (i, j) \in A \tag{10}$$

$$y_i \in \{0, 1\}, \forall i \in V \tag{11}$$

The objective function (1) aims to minimize the number of moves used by the mule to serve all nodes. Constraints (2) ensures that all nodes will be attended either by the mule's or by a neighbor in the mule's path. Constraints (3) and (4) guarantee that if one node is in the mule's path, at least one edge

must enter and at least one edge must leave this node. The set of constraints (5) ensures that the number of edges entering and leaving one node must be the same. Constraints (6) and (7) imposes the limits of edges entering and leaving a node i by the number of their neighbours ($N(i)$). The constraint (8) ensures that the Base Station belongs in the mule's path. The set of constraints (9) eliminate sub-cycles. Finally, constraints (10) and (11) define the domain of the variables.

4 Algorithms for Data Mule with Local View

In this section we present two strategies to deal with the Data Mule Routing Problem, when the data mule does not have a global view, i.e, a prior knowledge of the network as a whole. In the first one, the mule decides his path based on the number of uncovered nodes nearby the current sensor node. In the second, the mule decision is based on the computation of convex-hulls of the current sensor node [14].

4.1 Algorithms based on number of uncovered neighbours – AlgNUN

The data mule begins its path in the sensor node that represents the base station s and decides which will be the next sensor to be visited by using a greedy method that will be next explained. The edges traversed by the data mule forms a tree, where the nodes of the tree represent the sensors visited by the data mule.

We consider here that a sensor node can be in one of the following states: (i) *dominator*, when the data mule is or was located in the same position of this sensor node, (ii) *covered*, when the sensor is or was within the communication range of a *dominator*, indicating that it has already been served by the data mule, and (iii) *uncovered*, when it is not in any of the previous described states. Initially, except for the Base Station, all nodes are in the *uncovered* state. In the end of the algorithm, every node will be in either a *dominator* or a *covered* state.

The proposed algorithms, one for the data mule and the other for the regular sensor nodes are described in Algorithms 1 and 2. They use the following types of messages to decide about the data mule moving:

- *msg_request*: contains a request for number of uncovered neighbours, sent from data mule to a regular sensor node
- *msg_numernodes*: contains the number of uncovered neighbours, sent from regular sensor node to data mule
- *msg_serve*: informs that the node can already be served, sent from data mule to regular sensor node
- *msg_served*: informs that the node has been served, sent among regular sensor nodes

The data mule starts and finishes in a Base Station. Initially and upon reaching a sensor node u, the data mule sends the message *msg_serve* to all neighbours $N(u)$, indicating the node can be already served, updating their states to *covered* (lines 11-12 Algorithm 1). After that, the data mule also sends *msg_request* to them (line 13 Algorithm 1). Upon receiving that message (line 7 of Algorithm 2), a sensor node replies with the message *msg_numernodes* containing the number of neighbours in *uncovered* state (line 8 of Algorithm 2). The data mule, then, moves to the sensor node with the greatest number of uncovered neighbours (line 18 of Algorithm 1). When the mule arrives at a node u from a node v, if u is not in *dominator* state, it updates its state with *dominator* and the variable $parent(u)$ with v (lines 5-9 Algorithm 1). In this way, the data mule movement tree is being formed. When the data mule receives messages *msg_numernodes* containing only zeros, indicating that there is no neighbours in *uncovered* state, it moves to the parent of the current node. When this occurs in the Base Station, the data mule stops moving.

Algorithm 1: AlgNUM – Data Mule Algorithm

```
1  Variables:
2      parent ← ∅
3      states ← ∅
4  Upon reaching node u coming from node v
5  if u is not in dominator state then
6  |   states ← states \ {< u, covered >}
7  |   states ← states ∪ {< u, dominator >}
8  |   parent ← parent ∪ {< u, v >}
9  end
10 foreach v ∈ N(u) do
11 |   states ← states ∪ {< v, covered >}
12 |   Send msg_serve to v
13 |   Send msg_request to v
14 end
15
16 Upon receiving all msg_numbernodes of N(u)
17 if ∃ v ∈ N(u) | numbernodes(v) ≠ 0 then
18 |   Data Mule moves to sensor node t ∈ N(u) with the greatest
   |    numbernodes
19 else
20 |   if u is the Base Station then
21 |   |   Data Mule stop moving
22 |   else
23 |   |   v ← get_parent(parent, u)
24 |   |   Data Mule moves to v
25 |   end
26 end
```

Regarding the regular sensor node, when it receives the *msg_serve* from data mule, it sends the message *msg_served* to all neighbours (lines 4-6 Al-

gorithm 2). A sensor node when receives *msg_served*, decreases its variable containing the number of uncovered neighbours (lines 10-12 Algorithm 2). These steps allow for each sensor to keep the number of uncovered neighbours updated.

Algorithm 2: Regular Sensor Node u Algorithm

```
1  Variables:
2      uncoveredNodes ← |N(u)|
3  Upon receiving a message M
4  case M = msg_serve do
5  |  Send msg_served to all N(u)
6  end
7  case M = msg_request do
8  |  Send msg_numbernodes(uncoveredNodes) to mule
9  end
10 case M = msg_served do
11 |  uncoveredNodes– –
12 end
```

4.2 Algorithms based on convex-hull – AlgCH

The second proposed approach, AlgCH, follows the same steps of AlgNUM. However the criterion used to decide the data mule's movement is based on the convex-hull algorithm, next described.

Background

The convex hull of a set Q of points, called $CH(Q)$, is the smallest convex polygon P for which each point in Q is either on the boundary of P or in its interior. It is assumed that all points in Q are unique and that Q contains at least three no co-linear points [5]. An example of Q and the corresponding $CH(Q)$ are shown in Figure 2.

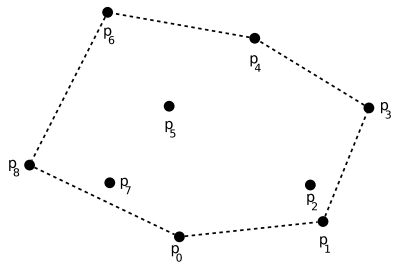

Fig. 2 $Q = \{p_0, p_1, \ldots, p_8\}$ and the corresponding $CH(Q)$ in dashed line

In this work, Graham's scan algorithm, presented in Algorithm 3, was used to calculate the convex-hull. Its running time is $O(n\ lg\ n)$, where n is the total number of points [5]. It uses a stack S to keep the points of the convex-hull. Initially, the algorithm chooses a point p_0 as the point with the lowest y-coordinate, picking the leftmost point (lowest x-coordinate) in case of a tie (line 2 Algorithm 3). Then, p_0 is pushed onto S (line 3 Algorithm 3). After that, the remaining $n-1$ nodes are sorted by the corresponding polar angle formed with p_0 in counterclockwise order, resulting in the ordered set $\langle p_1, p_2, ..., p_{n-1}\rangle$ (line 4 Algorithm 3). So, points p_1 and p_2 are also pushed onto S (lines 6-7 Algorithm 3). Note that in this algorithm, a point already pushed onto S can be popped from it depending on the next calculated polar angles. The popping decision is based on the polar angle formed between $\overrightarrow{p_i p_{\text{next}}}$ and $\overrightarrow{p_i p_{top}}$, that uses the procedures *Proc_top(S)* and *Proc_next(S)*, to obtain the first (top) and second elements (next) from the top of S, respectively. If the polar angle indicates a non left turn, *top* is popped from S. This procedure is repeat until a left turn is found, when then p_i is pushed onto S. An example of the algorithm execution is shown in Figure 3. Consider the scenario where p_0, p_1 are p_2 have already been pushed onto S, Figure 3(a). The new candidate point is p_3. As $p_{top} = p_2$ and $p_{\text{next}} = p_1$, the polar angle between $\overrightarrow{p_3 p_1}$ and $\overrightarrow{p_3 p_2}$ is calculated. Observe that in Figure 3(b) the angle indicates a nonleft turn, so p_2 is popped from S. After that, p_{top} and p_{next} are updated with p_1 and p_0, respectively. The polar angle between $\overrightarrow{p_3 p_0}$ and $\overrightarrow{p_3 p_1}$ is then calculated, indicating a left turn, Figure 3(c), that results in the pushing of p_3 onto S, Figure 3(d). The algorithm continues until all the nine points are verified, Figure 3(e).

Algorithm 3: Graham's Scan(Q) based on [5]

```
1  n ← |Q|
2  Let p_0 be the point in Q with the minimum y-coordinate, or the
     leftmost such point in caso of a tie
3  PUSH(p_0, S)
4  Let ⟨p_1, p_2, ..., p_{n-1}⟩ be the remaining points in Q sorted by polar angle
     in counterclockwise order around p_0;
5  Let S be an empty stack
6  PUSH(p_1, S)
7  PUSH(p_2, S)
8  for i = 3 to n − 1 do
9  |   while the angle formed by points Proc_next(S), Proc_top(S) and p_i
   |     makes a nonleft turn do
10 |   |   POP(S)
11 |   end
12 |   PUSH(p_i, S)
13 end
14 Set CH ← elements of stack S
15 return CH
```

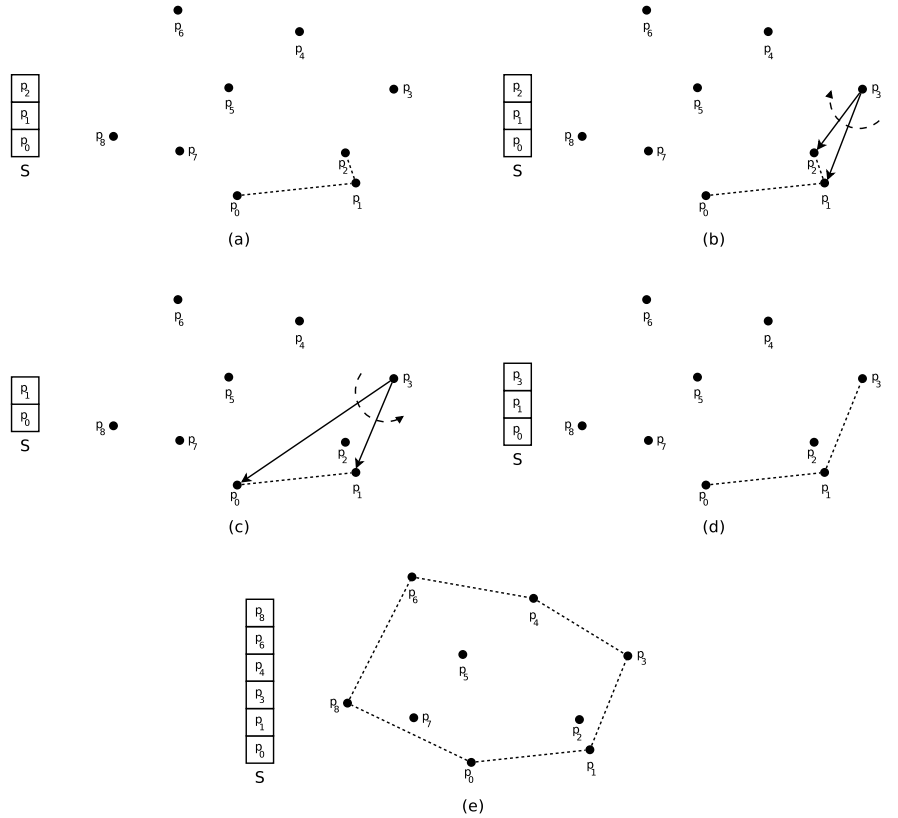

Fig. 3 Graham's scan first steps example

4.2.1 Data mule Algorithm

In AlgCH approach, regular sensor nodes execute also Algorithm 2, described in Subsection 4.1. Messages exchanged among the data mule and sensor nodes are the same previously used in AlgNUM. Only the data mule algorithm is modified, as presented in Algorithm 4. In both data mule algorithms, upon reaching a sensor node u, the date mule sends the message *msg_serve* to all nodes in $N(u)$, updating their status to *covered* (lines 11-14. However, in AlgCH before sending the next messages *msg_request*, the mule calculates the convex-hull of $N(u) \cup \{u\}$ (line 15 Algorithm 4), by using Algorithm 3. Here, the mule only sends *msg_request* to the regular sensors of the convex-hull (lines 16-18 Algorithm 4), which, in turn, reply with the number of uncovered neighbors. The mule, as in the previous algorithm, will move to the sensor that replied with the larger number of uncovered neighbours.

That approach aims at reducing the number of exchanged message between the data mule and sensor nodes. That happens because nodes inside the convex-hull do not receive *msg_request*. Another desired effect is the reduction

in the number of data mule movements. The idea is that the data mule could cover more sensor nodes, with less movements, if it moved on the boundary of the convex-hull.

Algorithm 4: AlgCH – Data Mule Algorithm

```
1  Variables:
2      parent ← ∅
3      states ← ∅
4      CHnodes ← ∅
5  Upon reaching node u coming from node v
6      if u is not in dominator state then
7          states ← states \ {< u, covered >}
8          states ← states ∪ {< u, dominator >}
9          parent ← parent ∪ {< u, v >}
10     end
11     foreach v ∈ N(u) do
12         states ← states ∪ {< v, covered >}
13         Send msg_serve to v
14     end
15     CHnodes ← Graham's Scan(N(u) ∪ {u})
16     foreach v ∈ CHnodes do
17         Send msg_request to v
18     end
19 end
20 Upon receiving all msg_numbernodes of CHnodes
21     if ∃ v ∈ CHnodes | numbernodes(v) ≠ 0 then
22         Data Mule moves to sensor node t ∈ CHnodes with the
             greatest numbernodes
23     else
24         if u is the Base Station then
25             Data Mule stop moving
26         else
27             v ← get_parent(parent, u)
28             Data Mule moves to v
29         end
30     end
31 end
```

4.3 Complexity Analysis

Table 3 presents memory and time asymptotic complexities both for the data mule and for the regular sensor node algorithms. The data mule stores two types of information: (i) a pair of integers (8 bytes) containing the node identification and its status, for each one of the n nodes of the WSN and (ii) a pair of integers (8 bytes) containing the node identification and its parent identi-

fication, for each one of the p visited nodes by the data mule, which forms the data mule movement tree. Thus, the memory complexity is $8(n+p)$ *bytes*. Because p can be equal to $n-1$ in the worst case, the memory asymptotic complexity is $\mathcal{O}(n)$ *bytes*.

Each regular sensor node u also stores two types of information: (i) an integer (4 *bytes*) containing the number of neighbours not covered yet and (ii) a list with $|N(u)|$ elements (each with 4 *bytes*) containing the neighbours identifications . Thus, the memory complexity is $4(|N(u)|+1)$ *bytes*. Because $|N(u)|$ can be equal to $n-1$ in the worst case, the asymptotic memory complexity is $\mathcal{O}(n)$ *bytes*.

Concerning the local time complexity, in AlgNUM the procedure with the biggest complexity is the one executed to select the node with the greatest number of uncovered neighbours, $\mathcal{O}(n)$, while, in AlgCH, the procedure to compute the convex hull is $\mathcal{O}(n \lg n)$. Regular sensor nodes only send messages and execute arithmetic operations, presenting local time complexity equal to $\mathcal{O}(1)$.

Table 3 Local Time and Memory Complexities

Complexity	**Mule**		**Regular Sensor** u
	AlgNUM	**AlgCH**	
Memory	$\mathcal{O}(n)$ bytes	$\mathcal{O}(n)$ bytes	$\mathcal{O}(n)$ bytes
Time	$\mathcal{O}(n)$	$\mathcal{O}(n \lg n)$	$\mathcal{O}(1)$

5 Computational experiments and Analysis

In order to evaluate the proposed algorithms, we used instances generated for the Close-enough Traveling Salesman Problem [18]. We selected ten instances that respect our problem assumptions: (i) nodes are located in an Euclidean plan, (ii) nodes have the same acting range, in our case, communication range, and (iii) the instances are connected graphs. The number of nodes of the selected instances varies from 100 to 1000.

We implemented the heuristics in two scenarios. In the first one, the data mule waits for an acknowledgment, from each sensor node to which it sent a *msg_serve*, before sending *msg_request*. In its turn, the sensor node waits for acknowledgements, from all sensors to which it sent *msg_served*, to send the corresponding acknowledgments to the data mule. Note that, in this scenario, all sent messages *msg_serve* and *msg_served* are delivered and processed, and, consequently, every sensor knows the correct number of covered nodes, when it receives a *msg_request*. The second scenario is the same described in Section 4.1 and 4.2, and no acknowledgment message is employed.

We also implemented the mathematical formulation and the algorithm for calculating a lower bound, as presented in sections 3.2 and 3.1. Those results are used as baseline to evaluate the quality of results and execution times of the proposed heuristics.

The heuristics were implemented in the programming language C++, and used MPI for message-passing. The mathematical formulation was implemented in the programming language C++ and used IBM ILOG CPLEX Optimizer v12.5.1 as mixed integer programming solver. The algorithm to calculate the lower bound (LB_3) calculation was implemented in C++ and used a graph library, LEMON[1]. Our tests were executed in a Intel Core i7 3.6 Ghz computer, with 16 GB of RAM and Linux Mint 18 as its operating system.

Table 4 presents the results obtained by the methods that have the global view of the network. It shows instance names in column **Inst.**; the lower bounds, in column $\boldsymbol{Sol_{LB_3}}$, found by using the Steiner Tree with $k = 3$, as presented in Lemma 4; the results obtained by the mathematical formulation are presented in columns, **LR**, linear relaxation on root node, $\boldsymbol{T_{LR}(s)}$, time to obtain this linear relaxation, $\boldsymbol{Sol_{Mat}}$, solution of entire formulation presented in Section 3.2, and the corresponding time, **T(s)**.

Table 4 Computational results – Global view

Inst.	$\boldsymbol{LB_3}$		**Mathematical Formulation**			
	$\boldsymbol{Sol_{LB_3}}$	**T(s)**	**LR**	$\boldsymbol{T_{LR}(s)}$	$\boldsymbol{Sol_{Math}}$	**T(s)**
kro100	4	0.01	3	0.01	4	1.31
rat195	4	0.07	3	0.02	4	4.27
team2_200	4	0.06	4	0.03	5	14.35
team3_300	32	0.09	19	0.07	74^{α}	17996.7
lin318	4	0.18	3.67	0.27	5^{α}	6012.51
rd400	6	0.28	5	0.97	6	7336.95
pcb442	6	0.38	4.14	0.88	6	37180.6
team6_500	3	0.66	3	2.20	3	225.29
dsj1000	6	3.00	4	2.50	8^{α}	24842.1
bonus1000	8	2.12	7.86	38.46	$22^{\alpha\beta}$	86400

$^{\alpha}$ the mathematical formulation used as input the best solution found by the four heuristics
$^{\beta}$ the optimal solution was not found in a time limit of 24 hours

It was not possible to prove the optimality of solution found for instance bonus1000, so only the feasible solution found was presented in the table. The calculated lower bound allows us to know the interval at which the best solution is, that can help us to evaluate the quality of solutions found by the heuristics. Note that the results showed that very good lower bounds were generated, matching with the optimal solutions in 5 of 10 instances and presenting better results concerning the linear relaxation of the mathematical formulation. Those results indicate that the lower bound algorithm gives good solutions, that can be also used as baseline to analyse heuristics results, in smaller times.

Table 5 summarizes the results obtained by the proposed heuristics. Column **Inst.** also presents instance names and the other columns present the

[1] LEMON – Library for Efficient Modeling and Optimization in Networks, available on https://lemon.cs.elte.hu

obtained results by **AlgNum** and **AlgCH** and the corresponding number of exchanged messages, **msgs**. Note that in **Case 2 – Without ACK**, the results are averages of 10 executions, because, as the algorithms do not use messages of acknowledgments, different solutions can be found at each execution.

Remark that the mathematical formulation has complete knowledge of the network while the heuristics are locality sensitive and have incomplete knowledge. Then, it was already expected that the results given by the heuristics were worse than the ones given by the mathematical formulation.

Table 5 Computational results – locality sensitive heuristics

Inst.	**Case 1 – With ACK**				**Case 2 – Without ACK**			
	Sol_{NUM}	**msgs**	Sol_{CH}	**msgs**	Sol_{NUM}	**msgs**	Sol_{CH}	**msgs**
kro100	6	11108	6	10538	10.0	6664.7	15.3	5852.4
rat195	4	43006	4	42108	6.4	23040.8	10.4	21795.9
team2_200	10	30562	12	29016	14.6	17515.6	18.4	15212.2
team3_300	74	21742	74	19556	94.2	13586.5	88.6	10876.1
lin318	8	94250	10	91874	13.2	51019.1	13.4	46772.0
rd400	14	112694	14	108482	22.8	62300.5	17.2	54981.5
pcb442	12	169950	12	165778	17.4	90869.9	17.8	85027.3
team6_500	6	312252	6	307818	12.0	164828.5	9.8	154971.2
dsj1000	8	837514	10	828926	17.8	431716.1	19.9	418260.2
bonus1000	22	450712	26	441438	36.4	238016.4	35.4	222859.8

Results presented in Table 5 show that the scenario **With ACK** produces better results, concerning the number of edges of the data mule path, than its counterpart. This occurs because in this scenario the data mule makes decisions based on a consistent state of the network. On the other hand, the number of exchanged messages increases thanks to ACK messages. In the scenario **With ACK**, *AlgNum* outperforms *AlgCH* in four instances and gives the same results in the other six, using, however, a greater number of messages. In the scenario **Without ACK**, *AlgNum* outperforms *AlgCH* in seven instances. However, *AlgCH* finds better solutions in three instances. Here, *AlgCH* also employs less messages than *AlgNum*.

Heuristics gave the optimal solutions in six cases (4 in AlgNUM and 2 in AlgCH) and they presented an average worsening percentage of 61.10% and 164.87% in **With ACK** and **Without ACK** cases, respectively, when compared with the mathematical formulation results. The results obtained by the proposed locality sensitive heuristics are good, since the mathematical formulation results were obtained in an unrealistic scenario where the data mule had the complete knowledge of the network.

Table 6 presents the average time for a data mule to make a decision about the next sensor node to be visited. This time includes the sending of *msg_request*, the receipt of *msg_numbernodes*, and in case of AlgCH, the time to calculate the convex hull, additionally. The results are averages of 10 executions.

Table 6 Maximum data mule local time

Inst.	Case 1 – With ACK		Case 2 – Without ACK	
	AlgNUM	AlgCH	AlgNUM	AlgCH
kro100	0.13	0.20	0.22	0.07
rat195	1.15	1.43	0.53	0.45
team2_201	0.36	0.73	0.86	0.85
team3_301	0.20	0.33	0.38	0.37
lin318	0.75	2.21	1.73	1.12
rd400	2.18	3.09	1.25	1.56
pcb442	1.72	2.27	1.73	1.60
team6_501	3.84	4.25	3.31	2.47
dsj1000	789.72	1213.97	290.25	242.11
bonus1001	9.13	7.91	6.30	3.62

It can be observed that in **With ACK** scenario, the calculus of the convex hull increases the time for the data mule to make a decision about the next sensor a lot. On the other hand, in **Without ACK** scenario, the opposite occurs, i.e., the data mule spends less time when executing the convex hull algorithm. It is due to the smaller number of *msg_request* messages sent in *AlgCH* than in *AlgNUM*, and consequently the reduction in waiting time for receiving the corresponding *msg_numbernodes* messages. It means that waiting for more messages is worse than calculating the convex hull in the tested instances.

6 Conclusion

This work dealt with the Data Mule Routing Problem, in a scenario where the data mule has only a local view of the WSN. Two locality sensitive heuristics *AlgNUM* e *AlgCH* were proposed and tested in instances of a related problem from the literature. They were compared in two different approaches, with and without the complete updating of the local knowledge, before making a decision about the next node to be visited. Experimental results showed that both approaches can be helpful depending on the main goal to be reached, i.e., to obtain the smallest data mule route, more messages must be exchanged by the sensor nodes and data mule, otherwise, the route can be a little worsened by reducing the number of exchanged messages. The proposed mathematical formulation and lower bounds showed to be important not only to define the problem when the data mule has the whole knowledge of the WSN, but to offer some parameters to analyse the quality of solutions given by the locality heuristics as well. The proposed heuristics presented good results in reasonable times when analysed considering those parameters.

Acknowledgements The authors acknowledges the support from CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) processes PDSEBEX 10380/13-2, PPVE 88887.122974/2016-00, and CNPq.

References

1. Alzoubi, K.M., Wan, P.J., Frieder, O.: New distributed algorithm for connected dominating set in wireless ad hoc networks. In: System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on, pp. 3849–3855. IEEE (2002)
2. Bin Tariq, M.M., Ammar, M., Zegura, E.: Message ferry route design for sparse ad hoc networks with mobile nodes. In: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing, MobiHoc '06, pp. 37–48. ACM, New York, NY, USA (2006)
3. Cheng, X., Huang, X., Li, D., Wu, W., Du, D.Z.: A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks. Networks **42**(4), 202–208 (2003)
4. Chlebk, M., Chlebkov, J.: Approximation hardness of dominating set problems in bounded degree graphs. Information and Computation **206**(11), 1264 – 1275 (2008)
5. Cormen, T.H., Stein, C., Rivest, R.L., Leiserson, C.E.: Introduction to Algorithms, 2nd edn. McGraw-Hill Higher Education (2001)
6. Das, B., Bharghavan, V.: Routing in ad-hoc networks using minimum connected dominating sets. In: Communications, 1997. ICC'97 Montreal, Towards the Knowledge Millennium. 1997 IEEE International Conference on, vol. 1, pp. 376–380. IEEE (1997)
7. Feige, U.: A threshold of ln n for approximating set cover. J. ACM **45**(4), 634–652 (1998)
8. Funke, S., Kesselman, A., Meyer, U., Segal, M.: A simple improved distributed algorithm for minimum cds in unit disk graphs. ACM Trans. Sen. Netw. **2**(3), 444–453 (2006)
9. Gandhi, R., Parthasarathy, S.: Fast distributed well connected dominating sets for ad hoc networks. Tech. rep., University of Maryland, CS-TR-4559 (2004)
10. Gendreau, M., Hertz, A., Laporte, G.: New insertion and postoptimization procedures for the traveling salesman problem. Operations Research **40**(6), 1086–1094 (1992)
11. Ghaffari, M.: Near-Optimal Distributed Approximation of Minimum-Weight Connected Dominating Set, pp. 483–494. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
12. Gudmundsson, J., Levcopoulos, C.: A fast approximation algorithm for tsp with neighborhoods and red-blue separation. Nordic Journal of Computing **6**, 6–469 (1997)
13. Guha, S., Khuller, S.: Approximation Algorithms for Connected Dominating Sets. Tech. Rep. 3660, UM Computer Science Department, University of Maryland (College Park, Md.) (1996)
14. Islam, K., Akl, S.G., Meijer, H.: A constant factor localized algorithm for computing connected dominating sets in wireless sensor networks. In: Parallel and Distributed Systems, 2008. ICPADS'08. 14th IEEE International Conference on, pp. 559–566. IEEE (2008)
15. Jang, H.C., Lien, Y.N., Tsai, T.C.: Rescue information system for earthquake disasters based on manet emergency communication platform. In: Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly, IWCMC '09, pp. 623–627. ACM, New York, NY, USA (2009)
16. Ma, M., Yang, Y.: SenCar: An Energy-Efficient Data Gathering Mechanism for Large-Scale Multihop Sensor Networks. IEEE Transactions on Parallel and Distributed Systems **18**(10), 1476–1488 (2007)
17. Ma, M., Yang, Y., Zhao, M.: Tour planning for mobile data gathering mechanisms in wireless sensor networks. IEEE Transactions on Vehicular Technology **62**(4), 1472–1483 (2013)
18. Mennell, W.K.: Heuristics for solving three routing problems: Close-enough traveling salesman problem, close-enough vehicle routing problem, sequence-dependent team orienteering problem. Ph.D. thesis, University of Maryland (College Park, Md.), College Park, Maryland, USA (2009)
19. Ngai, E.C.H., Liu, J., Lyu, M.R.: An adaptive delay-minimized route design for wireless sensor-actuator networks. IEEE Trans. Vehicular Technology **58**(9), 5083–5094 (2009)
20. Peleg, D.: Distributed Computing: A Locality-sensitive Approach. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2000)
21. Puccinelli, D., Haenggi, M.: Wireless sensor networks: applications and challenges of ubiquitous sensing. IEEE Circuits and Systems Magazine **5**, 2005 (2005)

22. Rao, J., Wu, T., Biswas, S.: Network-Assisted Sink Navigation Protocols for Data Harvesting in Sensor Networks. 2008 IEEE Wireless Communications and Networking Conference pp. 2887–2892 (2008)
23. Sahin, C.S., Urrea, E., Uyar, M.U., Conner, M., Hokelek, I., Bertoli, G., Pizzo, C.: Uniform distribution of mobile agents using genetic algorithms for military applications in manets. In: Military Communications Conference, 2008. MILCOM 2008. IEEE, pp. 1–7. IEEE (2008)
24. Sharma, S., Bansal, R.K., Bansal, S.: Issues and challenges in wireless sensor networks. In: Machine Intelligence and Research Advancement (ICMIRA), 2013 International Conference on, pp. 58–62. IEEE (2013)
25. Stojmenovic, I., Seddigh, M., Zunic, J.: Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. IEEE Trans. Parallel Distrib. Syst. **13**(1), 14–25 (2002)
26. Sugihara, R., Gupta, R.K.: Speed control and scheduling of data mules in sensor networks. ACM Transactions on Sensor Networks **7**(1), 4:1–4:29 (2010)
27. Sugihara, R., Gupta, R.K.: Path planning of data mules in sensor networks. ACM Transactions on Sensor Networks **8**(1), 1:1–1:27 (2011)
28. Wichmann, A., Chester, J., Korkmaz, T.: Smooth path construction for data mule tours in wireless sensor networks. In: GLOBECOM, pp. 86–92 (2012)
29. Wu, J., Li, H.: On calculating connected dominating set for efficient routing in ad hoc wireless networks. In: Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, DIALM '99, pp. 7–14. ACM, New York, NY, USA (1999)
30. Xing, G., Wang, T., Xie, Z., Jia, W.: Rendezvous Planning in Wireless Sensor Networks with Mobile Elements. IEEE Transactions on Mobile Computing **7**(12), 1430–1443 (2008)
31. Yuan, B., Orlowska, M., Sadiq, S.: On the Optimal Robot Routing Problem in Wireless Sensor Networks. IEEE Transactions on Knowledge and Data Engineering **19**(9), 1252–1261 (2007)
32. Zhao, M., Gong, D., Yang, Y.: Network cost minimization for mobile data gathering in wireless sensor networks. IEEE Transactions on Communications **63**(11), 4418–4432 (2015)
33. Zhao, W., Ammar, M.: Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks. The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems, 2003. FTDCS 2003. Proceedings. pp. 308–314 (2003)
34. Zhao, W., Ammar, M., Zegura, E.: Controlling the mobility of multiple data transport ferries in a delay-tolerant network. In: Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies., vol. 2, pp. 1407–1418. IEEE (2005)