Improved solution to data gathering with mobile mule

Yoad Zur and Michael Segal

September 17, 2019

Communication Systems Engineering Department,
School of Electrical and Computer Engineering,
Ben-Gurion University of the Negev, Beer-Sheva, Israel
Emails: yoadzu@post.bgu.ac.il, segal@bgu.ac.il

Abstract

In this work we study the problem of collecting protected data in ad-hoc sensor network using a mobile entity called MULE. The objective is to increase information survivability in the network. Sensors from all over the network, route their sensing data through a data gathering tree, towards a particular node, called the *sink*. In case of a failed sensor, all the aggregated data from the sensor and from its children is lost. In order to retrieve the lost data, the MULE is required to travel among all the children of the failed sensor and to re-collect the data. There is a cost to travel between two points in the plane. We aim to minimize the MULE traveling cost, given that any sensor can fail. In order to reduce the traveling cost, it is necessary to find the optimal data gathering tree and the MULE location. We are considering the problem for the unit disk graphs (UDG) and Euclidean distance cost function. We propose a primal-dual algorithm that produces a $(20 + \varepsilon)$-approximate solution for the problem, where $\varepsilon \to 0$ as the sensor network spreads over a larger area. The algorithm requires $O\left(n^3 \cdot \Delta\left(G\right)\right)$ time to construct a gathering tree and to place the MULE, where $\Delta\left(G\right)$ is the maximum degree in the graph and n is the number of nodes.

Keywords: WSN · MULE · MWCDS · Primal-Dual · UDG

1 Introduction

Consider an Ad-Hoc sensor network randomly embedded in the plane. An example application is the use of sensors scattered in forests using aircraft, to give an indication when a fire starts. The main problem in WSN is long-term survivability. Because of the limited energy of the sensor, its lifetime depends on its energy consumption. A standard solution for efficient utilization of energy is to build a data gathering tree in the network. In this method, the data is aggregated in order to reduce the number of messages each sensor transmits. The sensors create a logical construction of a tree in the network, and all the data routed to the sink according to that tree. The basic principle of the data aggregation is that each sensor waits to aggregate the data from all its children before it sends its own message to its father. When a sensor failure occurs, all the aggregated data from the sensor and its children are lost. Also, if the sensor is not a leaf in the tree, its entire subtree is disconnected. It takes a long time to recognize the fault, and for the network to reconstruct a new gathering tree. Here comes the use of data MULE to retrieve the lost data. Data MULE is a mobile unit provided with extended computing and storage capabilities, and short-range wireless communication.

In order to gather data from the children of a failed sensor, the MULE should travel and approach each child. After visiting all of the children, the MULE returns to its starting point. There is a cost to travel between two points. Given a graph of gathering sensors network, we would like to find the data gathering tree and the position of the MULE, which minimizes the traveling cost, while each sensor could fail. It

is common to model wireless sensor networks as a Unit Disk Graph (UDG), where the transmission range of the sensors is defined as one unit. The sensors are the nodes, and between each pair of sensors that can communicate directly with each other, there will be an edge in the graph. The MULE problem was previously defined by Crowcroft et al. [3] and Yedidsion et al. [14].

In this work, we focused primarily on the study of the problem on general unit disk graphs for Euclidean distance cost function, and only one sensor can fail at a time.

The paper is organized as follows. In the next section we present the previous related work. In Section 3 we define our problem and model, and introduce the main subjects that our method is based on. In Section 4 we show reductions of the problem which we use later. We present our algorithm and analyze its performance, in Section 5. In Section 6 we present simulation results of our algorithm, and conclude in Section 7.

2 Related work

In the last years, some research has been started to evaluate how Data MULEs can benefit the performance of wireless sensor networks. Two major approaches have developed in this area. The first approach is the regular use of data MULEs in order to pass data between remote components in the network [10, 8, 11]. The second approach is using data MULEs in order to increase information survivability in the network. The main idea is to use MULEs to amend failures in the network. This is where our work takes place. Crowcroft et al. [3] use this approach to deal with the problem of efficient data recovery using the data MULEs approach. They consider a sensor network that uses a data gathering tree. The main idea is to use MULEs in order to retrieve lost data caused by failed sensors by visiting the children of the failed sensors. They define the problem as "$(\alpha, \beta) - Mule$", when α is the amount of simultaneously failed sensors and β is the number of MULEs in the network. They study several aspects of the problem and suggest a variety of solutions to the problem on the following graph topologies: complete graph and UDG (line, random line, and grid). Yedidsion et al. [14] also study the same problem of data gathering in sensor networks using MULE. They consider the problem when there is one MULE in the network, and only one sensor can fail at the same time. They study the problem for several topologies, such as UDG on a line and general UDG. Further, they also consider a failure probability for each sensor and study the problem for a complete graph. For the UDG on a line, they present an $O(n)$ time algorithm that solves the problem. For a general UDG graph, they present an algorithm and two approaches to analyze its approximation ratio, with a tradeoff between runtime and approximation. The first approach achieves $(57 + \epsilon)$ approximation and takes $O(n^3 \log n)$ time, where ϵ comes from a TSP calculation (according to [9]) for the approximation ratio. The second approach reduces operations and achieves $O(n^2 \log n)$ time. However, it pays with a worse approximation of $(114 + \epsilon)$. Ashur [1] continued their study on a general UDG, and came up with a very large constant-factor approximation algorithm for the problem, which requires $O(n \log n)$ runtime.

Like Yedidsion et al. [14] and Ashur [1], we will also focus on the MULE problem for a general UDG graph, when there is one MULE, and only one sensor can fail at a time. We use a different approach to solve the problem, and propose a primal-dual algorithm that achieves a $(20 + \varepsilon)$-approximation for the MULE problem but has a higher runtime of $O(n^3 \cdot \Delta(G))$. We note that $\Delta(G)$ is the maximum degree in the graph, n is the number of nodes, and $\varepsilon \to 0$ as the sensor network spreads over a larger area.

3 Preliminaries and Model

3.1 Problem definition

Definition 1 (Data gathering tree)**.** Data gathering tree $T = (V, E_T)$ is a directed spanning tree in graph $G = (V, E)$. Consider a spanning tree in G and a root node r, T is the rooted directed version of the spanning tree, where from each node in the tree there is a directed path to r.

In this work, we study the MULE problem for UDG, where there is one MULE and only one sensor failure at a time, in the network. We will refer to this problem as "$(1,1) - MULE - UDG$", and it is defined as follows: Consider a network of sensors embedded in the plane with the same transmission range. Let $G = (V, E)$ be a connected UDG that models the WSN, when the sensors are nodes, and an edge is defined between a pair of nodes only if they are within the transmission range of each other. Let T be a data gathering tree which spans all the nodes in G. T is rooted at node r, and sensing data propagates from leaf nodes to the root. One MULE is located at some node on the graph. Let m be the MULE location. The MULE can travel between a set of nodes, where the cost function of a tour t (referred to as $c(t)$) is its total *Euclidean distances*. Let $\delta(v, T)$ be the set of children that v has in tree T. When sensor v_f fails, all data gathered from its children is lost, and the MULE must travel through all of its children, in order to retrieve the lost data. The MULE performs a circular tour that goes through all the nodes $\delta(v_f, T)$ and at the end returns to the starting position m. Let $t(m, \delta(v_f, T))$ be the shortest circular tour through all the nodes in $\{m\} \cup \delta(v_f, T)$, that the MULE has to take. The objective will be to minimize the overall MULE tour, regardless of which sensor fails. Formally, the problem is defined as follows:

Definition 2 ($(1,1) - MULE - UDG$ problem)**.** Given a connected UDG graph denoted by $G = (V, E)$, our goal is to find the MULE location m and the data gathering tree T, which minimizes the objective function:

$$\min_{m, T} \left[\sum_{v_f \in V} [c(t(m, \delta(v_f, T)))] \right] .$$

3.2 The primal-dual method

The field of combinatorial optimization problems has been heavily influenced by the field of Linear Programming (LP) since many combinatorial problems can be described as linear programming problems. The *primal-dual* method was first introduced by Dantzig, Ford, and Fulkerson [4] in 1956, in order to find an exact solution for linear programming problems. The principle of the *primal-dual* method can also be useful for finding an approximate solution in polynomial time for NP-hard optimization problems, and this method is called "*The Primal-Dual Method for Approximation Algorithms*". The main idea is to construct a feasible solution for the LP problem (referred as the "*Primal LP Problem*") from scratch, using a related LP problem (referred as the "*Dual LP Problem*") that guides us during the construction of the feasible solution. In our case, the relationship between the *primal* and the *dual* problems is such that, the *primal* solution serves as an upper bound, while the *dual* solution serves as a lower bound for the optimal solution OPT. The performance of a *primal-dual* approximation algorithm is measured by the ratio between the *primal* and the *dual* solutions it finds. In 1995, Goemans and Williamson [6] proposed a general technique for designing approximation algorithms based on this method. Their technique produces 2-approximation algorithms for a broad set of graph problems that run in $O(n^2 \log n)$ time. Their technique had a significant impact on our research. For more details about the *primal-dual* method, we will refer the reader to the papers [6, 12, 7].

3.3 Minimum-weighted connected dominating set problem

For a later use, we want to define the term MIS and the problem of MWCDS.

Given a graph $G = (V, E)$, a set $S \subseteq V$ will be called an *Independent Set* (IS), if for each pair of nodes $u, v \in S$ there is no edge $(u, v) \in E$. The set will also be called a *Maximal Independent Set* (MIS), if it is an IS and there is no node $v \in V \setminus S$ which can be added to the set while it will still remain independent.

Consider the graph $G = (V, E)$, a set of nodes $S \subseteq V$ will be called a *Dominating Set* (DS), if each node $v \in V$ is either in the set or is a neighbor of any node in the set. The DS with the minimum cardinality is called a *Minimum Dominating Set* (MDS). A set of nodes $S \subseteq V$ will be called *Connected Set* (CS) if its induced graph in G is connected. A set of nodes $S \subseteq V$ will be called *Connected Dominating Set* (CDS) if S is both dominating set and connected. The CDS with the minimum cardinality is called a *Minimum Connected Dominating Set* (MCDS). Consider a weight function for each node in graph G, the DS with the

minimum total node weights is called a *Minimum-Weighted Dominating Set* (MWDS) and the CDS with the minimum total node weights is called a *Minimum-Weighted Connected Dominating Set* (MWCDS).

Definition 3 (MWCDS problem). Given a connected graph denoted by $G = (V, E)$ and a node weight function $w(v) : V \to R^+$, our goal is to find a subset of nodes $S \subseteq V$, which is CDS in G and minimizes the objective function:

$$\min_{S \subseteq V} \left\{ \sum_{v \in S} w(v) \right\} .$$

Finding MCDS is an NP-complete problem for UDG ([2]). Hence the more general case of MWCDS is also an NP-complete problem. In 2010 Erlebach and Matus [5] presented a polynomial-time algorithm that achieves approximation of $(4 + \epsilon)$ for MWDS, and they combine it with a known algorithm for node-weighted Steiner trees to achieve a 7.875-approximation for MWCDS in UDG graphs. In addition, independently, in 2011 Zou et al. [15] also introduced a polynomial-time algorithm that achieves $(4 + \varepsilon)$-approximation to the MWDS problem on UDG. Due to the method they were based on, there is a trade-off between the approximation performance ratio and the runtime. So they only proved that the runtime was polynomial, but they did not analyze it accurately.

We wanted to put more emphasis on a reasonable runtime, and therefore in this work, we chose to use a different method than the others: we will use the *Primal-Dual* method.

4   Reduction of The Problem

4.1   Reduction to an approximate MWCDS problem

The MULE problem we study seems to be an NP-hard problem, although it has not yet been proven. It is interesting to note that with some reasonable assumptions, the $(1,1) - MULE - UDG$ problem can be reduced to the problem of finding a MWCDS, when the node weight is proportional to its *Euclidean distance* from the MULE. We assume that the MULE has a fixed transmission range of R_M, within the range $0 < R_M < 0.3 \cdot R$, where R is the sensor's transmission range. In our case, for UDG, $R = 1$ and the range will be $0 < R_M < 0.3$. In other words, the MULE does not need to reach the exact location of the sensor, but it needs to get close enough to communicate with it.

Let m be any location of the MULE, and T be some gathering tree. The MULE's tour for a failure $t(m, \delta(v, T))$ of node v can be represented as a series of nodes according to their order $\{m, u_1, u_2, u_3, \ldots, u_{|\delta(v,T)|}, m\}$. We will consider the tour as three separate parts. $LWF(v, T)$ ("Long Walk Forward") is the cost of the MULE's walk from node m to node u_1, for the failure of node v in gathering tree T. $LWB(v, T)$ ("Long Walk Backward") is the cost of the MULE's walk from node $u_{|\delta(v,T)|}$ to node m. $WAC(v, T)$ ("Walking Among Children") is the cost of the MULE's walk from node u_1 to node $u_{|\delta(v,T)|}$, through the rest of v's children $u_2, u_3, \ldots, u_{|\delta(v,T)|-1}$ in tree T.

Let $Cost(T, m)$ be the total tours cost $\sum_{v_f \in V} c(t(m, \delta(v_f, T)))$ for a MULE located at m and for gathering tree T. Let us denote the BackBone (BB) of gathering tree T to be the induced undirected subgraph of T, which contains all the non-leaf nodes, that is, all nodes with degree greater than one. The BackBone is denoted by $T_{BB} = (V_{BB}, E_{BB})$, where V_{BB} is the set of the BackBone's nodes and E_{BB} is its set of edges. Let OPT_{T^*} be the cost of the optimal solution for the $(1,1) - MULE - UDG$ problem, and m^* be the optimal MULE location, where $T^* = (V, E_T^*)$ is the data gathering tree of the solution. That is, $OPT_{T^*} = Cost(T^*, m^*)$. Let V_{BB}^* be the BackBone nodes of T^*. We denote by $dist\{m, v\}$ the *Euclidean distance* between node v and the MULE location m.

Proposition 4. *If we consider the MULE's transmission range as a constant in the range $0 < R_M < 0.3$, then for any data gathering tree T and for each node $v \in V$, we can bound $WAC(v, T) \leq C_1$, where $C_1 = (1 + R_M)(1 + \pi(1 + \lceil (1-R_M)/2R_M \rceil))$.*

Proof. Note that all the children of any node are scattered over a fixed area due to the UDG property, a disk area of radius 1 (i.e. the node transmission area). One extreme possibility for the MULE's tour will be to
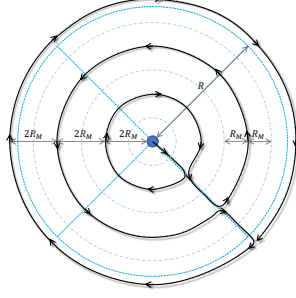
Figure 1: Bounded walk among children of a failed sensor.

reach a failed node with precisely one child. In this case, the MULE has no need to walk in the transmission area around the node, and therefore we have $WAC(v,T) = 0$. As the nodes density increases in the graph, the children of a node are increasingly filling the transmission area around it, and the MULE will have to walk a lot more to reach all of them. Therefore, the extreme case on the other side will be when the MULE has to walk **all over the transmission area** of the failed node, in order to reach all of its children. Since we assumed that the MULE has a fixed transmission range R_M, we want to show that there is a fixed-length walk during which the MULE approaches each child at a distance of R_M.

The description of the walk is as follows. First, the MULE reaches the center of the area, where the failed node is located. From there, it moves in increasing circles, until full coverage of the area (as described in Figure 1). The first circle is with radius $2R_M$, and for all the rest, each circle has a radius that grows by $2R_M$ from its predecessor. Each time the MULE completes an entire circle, it moves $2R_M$ units away in the direction of the radius and starts the walk of the next circle. In order to cover the whole transmission area, the amount of circles required is $\lceil (R-R_M)/2R_M \rceil$.

Calculating the total length of the MULE walk consists of walking in circles $\sum_{i=1}^{\lceil (R-R_M)/2R_M \rceil} 2\pi \cdot (i \cdot 2R_M)$ and walking between circuits $\lceil (R-R_M)/2R_M \rceil \cdot 2R_M$. Let us define $N_R \triangleq \lceil (R-R_M)/2R_M \rceil$, we get:

$$
\begin{aligned}
WAC(v,T) &\leq N_R \cdot 2R_M + \sum_{i=1}^{N_R} 2\pi \cdot (i \cdot 2R_M) \\
&= N_R \cdot 2R_M + \frac{N_R}{2}(2\pi \cdot 2R_M + 2\pi \cdot N_R \cdot 2R_M) \\
&= 2R_M \cdot N_R (1 + \pi(1 + N_R)) \\
&= 2R_M \cdot \lceil (R-R_M)/2R_M \rceil (1 + \pi(1 + \lceil (R-R_M)/2R_M \rceil)) \\
&\leq (1 + R_M)(1 + \pi(1 + \lceil (1-R_M)/2R_M \rceil)) \\
&= C_1 .
\end{aligned}
$$

The second inequality derives from the rounding up operation and since $R = 1$. □

Lemma 5. *Let $(2 \cdot dist\{m,v\} + C)$ be the weight of node v, where C is a positive constant dependent on R_M. Then $Cost(T,m)$ can be bounded from **above** by the total weight of $T's$ BackBone nodes.*

Proof. Since a leaf node has no children in the tree T, the MULE does not go on tour for the failure of a leaf node. Therefore, the only nodes for which the tour cost $c(t(m, \delta(v_f, T)))$ is greater than 0 are the BackBone nodes of T. So, we can write:

$$
Cost(T,m) \triangleq \sum_{v_f \in V} c(t(m, \delta(v_f, T))) = \sum_{v \in V_{BB}} c(t(m, \delta(v, T))) . \tag{1}
$$

The MULE's tour consists of 3 different walks, $LWF(v,T)$, $LWB(v,T)$ and $WAC(v,T)$. Note that from the triangle inequality, and since a child of node v can be at most one unit away from v, we get: $LWF(v,T) \leq dist\{m,v\} + 1$ and $LWB(v,T) \leq dist\{m,v\} + 1$. Combining this with Proposition 4, we

5

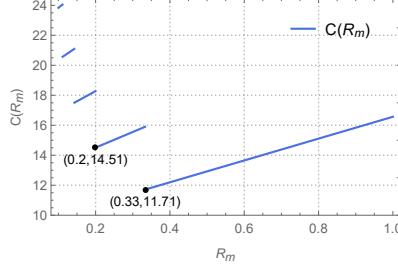Figure 2: Constant C as a function of R_M.

will get:

$$
\begin{aligned}
c\left(t\left(m, \delta\left(v, T\right)\right)\right) = & \ LWF\left(v, T\right) + WAC\left(v, T\right) + LWB\left(v, T\right) \\
\leq & \ \left(dist\left\{m, v\right\} + 1\right) + C_1 + \left(dist\left\{m, v\right\} + 1\right) \\
= & \ 2 \cdot dist\left\{m, v\right\} + 2 + C_1 \ .
\end{aligned}
\tag{2}
$$

Let us define $C \triangleq 2 + C_1$. Thus, in conclusion:

$$
\begin{aligned}
Cost\left(T, m\right) \overset{(1)}{=} & \ \sum_{v \in V_{BB}} c\left(t\left(m, \delta\left(v, T\right)\right)\right) \\
\overset{(2)}{\leq} & \ \sum_{v \in V_{BB}} \left(2 \cdot dist\left\{m, v\right\} + C\right) \ ,
\end{aligned}
\tag{3}
$$

where, $C = 3 + R_M + \pi\left(1 + R_M\right)\left(1 + \left\lceil (1-R_M)/2R_M \right\rceil\right)$ (Figure 2). $\qquad\square$

Lemma 6. *Let $\left(2 \cdot dist\left\{m, v\right\} - 2.6\right)$ be the weight of node v. Then $Cost\left(T, m\right)$ can be bounded from **below** by the total weight of $T's$ BackBone nodes.*

Proof. Note that the shortest tour to a particular failed node v is in case the node has exactly one child in T, which is located as close as possible to the MULE. Since this is only one child, then $WAC\left(v, T\right) = 0$. We will distinguish between two possible cases: when the MULE is located within the transmission area of the failed node v, and when it is located outside it. In the first case, $dist\left\{m, v\right\} \leq 1$ and the shortest possible tour will have a cost of 0 (in case the MULE is located such that the only child is within its transmission range). So we have $LWF\left(v, T\right) = LWB\left(v, T\right) = 0 \geq dist\left\{m, v\right\} - 1.3$. In the second case, where $dist\left\{m, v\right\} > 1$, the location of the child that minimizes its distance from the MULE will be directly between the node v and the MULE, at the edge of v's transmission range (exactly one unit away from v). This implies that $LWF\left(v, T\right) = LWB\left(v, T\right) = dist\left\{m, v\right\} - 1 - R_M$. Recall that $R_M < 0.3$. Therefore we can bound the MULE's tour from below, as follows:

$$
c\left(t\left(m, \delta\left(v, T\right)\right)\right) = LWF\left(v, T\right) + WAC\left(v, T\right) + LWB\left(v, T\right) > 2 \cdot \left(dist\left\{m, v\right\} - 1.3\right) \ .
\tag{4}
$$

Combine it with Equation 1 of Lemma 5, we get:

$$
Cost\left(T, m\right) \overset{(1)}{=} \sum_{v \in V_{BB}} c\left(t\left(m, \delta\left(v, T\right)\right)\right) \overset{(4)}{>} \sum_{v \in V_{BB}} \left(2 \cdot dist\left\{m, v\right\} - 2.6\right) \ .
$$

$\qquad\square$

Theorem 7. *Consider the optimal solution for the MWCDS problem [3] for node weights $\left(2 \cdot dist\left\{m^*, v\right\} + C\right)$, where C is a positive constant dependent on R_M and m^* is the optimal MULE location for the $(1, 1) - MULE - UDG$ problem. Let V_{BB}^* be the BackBone nodes of the optimal gathering tree. If the condition $\underset{v \in V_{BB}^*}{Average}\left[dist\left\{m^*, v\right\}\right] > 1.3$ is met, then a data gathering tree whose BackBone consists of the optimal solution to this specific MWCDS problem is an $(1 + \varepsilon)$-approximate solution for the $(1, 1) - MULE - UDG$ problem, where as the area of the BackBone increases, the smaller the ε (i.e. $\varepsilon \to 0$ for $\underset{v \in V_{BB}^*}{Average}\left[dist\left\{m^*, v\right\}\right] \to \infty$).*

6

Proof. We will denote OPT_{cds^*} to be the total weight of the optimal solution for the MWCDS problem, where cds^* is the optimal CDS itself, which achieves this weight. If the nodes weights are $(2 \cdot dist\{m^*, v\} + C)$, then according to the problem definition [3], we have:

$$OPT_{cds^*} = \sum_{v \in cds^*} (2 \cdot dist\{m^*, v\} + C) \ . \tag{5}$$

From the other side, of the $(1,1) - MULE - UDG$ problem, let us define the weight of a gathering tree T (referred as $Weight_{BB}(T)$), to be the total weight of its BackBone nodes.

We want to claim that the BackBone of T is a CDS. Therefore we are required to show that the BackBone is both dominating and connected. T is a spanning tree in G by definition [1]. Thus, it is connected and contains all the nodes. Since T is connected, each node in T is either a leaf or a BackBone node, and if it is a leaf, then it has a neighbor in the BackBone. So, the BackBone is a DS. T is connected by definition [1]. Therefore, if we will prune all of its leaves it will still remain connected. Hence, the BackBone nodes are also a CDS.

We assume that the root node is not a leaf. Otherwise, we can always replace T^* with such a tree, without affecting the approximation ratio. Let r and v be the root node and one of its neighbors, respectively. If r is a leaf, we can replace v to be the root instead. Thus we will create a new tree T', for which $LWF(v, T') = LWB(v, T') = 0$ and $WAC(v, T') \le WAC(v, T^*) + 4$. The worst case of adding 4 to the total tour cost of T' does not affect the approximation ratio.

The BackBone nodes of T is a CDS, and the root is one of these nodes. If the total weight of the BackBone is the weight of the tree, then it can be said that, finding a tree with the lowest weight is equivalent to finding a MWCDS in the graph. Formally, the following equality holds:

$$
\begin{aligned}
\min_T \{Weight_{BB}(T)\} &= \min_T \left\{ \sum_{v \in V_{BB}} (2 \cdot dist\{m^*, v\} + C) \right\} \\
&= \min_{S \subseteq V} \{Weight(S)\} \\
&\quad s.t\ S\ is\ a\ CDS \\
&= \sum_{v \in cds^*} (2 \cdot dist\{m^*, v\} + C) \ .
\end{aligned} \tag{6}
$$

A gathering tree with the lowest weight has a lower weight than any other tree, in particular T^*. Note that these are not necessarily the same gathering trees. Therefore:

$$
\begin{aligned}
\min_T \{Weight_{BB}(T)\} &= \min_T \left\{ \sum_{v \in V_{BB}} (2 \cdot dist\{m^*, v\} + C) \right\} \\
&\le Weight_{BB}(T^*) \\
&= \sum_{v \in V_{BB}^*} (2 \cdot dist\{m^*, v\} + C) \ .
\end{aligned} \tag{7}
$$

From Lemma 5 and Lemma 6 we have seen that OPT_{T^*} can be bounded as follows:

$$\sum_{v \in V_{BB}^*} (2 \cdot dist\{m^*, v\} - 2.6) < OPT_{T^*} \le \sum_{v \in V_{BB}^*} (2 \cdot dist\{m^*, v\} + C) \ .$$

Now, we will calculate how much the upper bound is greater than the optimal solution.

$$
\begin{aligned}
\frac{\sum_{v \in V_{BB}^*} (2 \cdot dist\{m^*, v\} + C)}{OPT_{T^*}} &< \frac{\sum_{v \in V_{BB}^*} (2 \cdot dist\{m^*, v\} + C)}{\sum_{v \in V_{BB}^*} (2 \cdot dist\{m^*, v\} - 2.6)} \\
&= \frac{\sum_{v \in V_{BB}^*} 2 \cdot dist\{m^*, v\} + |V_{BB}^*|C}{\sum_{v \in V_{BB}^*} 2 \cdot dist\{m^*, v\} - |V_{BB}^*|2.6} \\
&= 1 + \frac{1}{\frac{2}{(C+2.6)} \cdot \left(\frac{\sum_{v \in V_{BB}^*} dist\{m^*, v\}}{|V_{BB}^*|} - 1.3 \right)} \\
&= 1 + \frac{1}{\frac{2}{(C+2.6)} \cdot \left(\underset{v \in V_{BB}^*}{Average}[dist\{m^*, v\}] - 1.3 \right)} \\
&= 1 + \varepsilon \ ,
\end{aligned} \tag{8}
$$

where, $\varepsilon \triangleq \left(\frac{2}{(C+2.6)} \cdot \left(\underset{v \in V_{BB}^*}{Average} [dist \{m^*, v\}] - 1.3 \right) \right)^{-1}$, and hence $\varepsilon \to 0$ for $\underset{v \in V_{BB}^*}{Average} [dist \{m^*, v\}] \to \infty$.

That is, the ε error decreases as the average distance of the BackBone nodes from the MULE, increases. Also, as the graph will be deployed over a wider area. We should note that for correct approximation, the condition $\underset{v \in V_{BB}^*}{Average} [dist \{m^*, v\}] > 1.3$ must be met.

Finally, we will summarize everything in one equation:

$$
\begin{aligned}
OPT_{cds^*} &\overset{(5)}{=} \sum_{v \in cds^*} (2 \cdot dist \{m^*, v\} + C) \\
&\overset{(6)}{=} \min_T \left\{ \sum_{v \in V_{BB}} (2 \cdot dist \{m^*, v\} + C) \right\} \\
&\overset{(7)}{\leq} \sum_{v \in V_{BB}^*} (2 \cdot dist \{m^*, v\} + C) \\
&\overset{(8)}{<} (1 + \varepsilon) OPT_{T^*} .
\end{aligned}
\tag{9}
$$

$\square$

4.2 The primal-dual method for MWCDS problem

In this section, we will see how the $Primal - Dual$ method can assist us in finding an approximate solution to the MWCDS problem with which we are dealing.

Consider the following Integer Program (IP) problem:

$$
\begin{aligned}
\min_{\bar{x}} \quad & \sum_{v \in V} w(v) \cdot x_v \\
s.t. \quad & \sum_{v \in N(S)} x_v \geq f(S) \quad \forall S : S \subseteq V, S \neq \emptyset \qquad (MWCDS - IP) . \\
& x_v \in \{0, 1\}
\end{aligned}
$$

Let $\bar{x}$ be a vector of size $|V|$, where x_v are its components, and let $w(v)$ be some weight function $w(\cdot) : V \to R^+$. We denote by $N(S)$ the set of nodes which neighbors to set $S \subseteq V$. Formally: $N(S) \triangleq \{u : \exists e_{u,v} \in E : v \in S, u \in V \setminus S\}$. $f(S)$ is an indicator function which returns 0 if set $S \subseteq V$ is a DS. Formally:

$$
f(S) \triangleq \begin{cases} 0 & for \ N(S) \cup S = V \\ 1 & otherwise \end{cases} .
\tag{10}
$$

This IP optimization problem, denoted by $MWCDS - IP$, is to find a feasible vector $\bar{x}$ which minimizes the sum $W_{\bar{x}} \triangleq \sum_{v \in V} w(v) \cdot x_v$.

We will denote by $MWCDS - LP$ the Linear Program (LP) relaxation of the $MWCDS - IP$ problem, where we relax the integrality constraint $x_v \in \{0, 1\}$ to be $x_v \geq 0$. This is our *primal* problem.

The *dual* problem for $MWCDS - LP$ will be referred to as $MWCDS - D$, and is defined as follows:

$$
\begin{aligned}
\max_{\bar{y}} \quad & \sum_{S \subseteq V : S \neq \emptyset} f(S) \cdot y_S \\
s.t. \quad & \sum_{S \subseteq V : S \neq \emptyset, v \in N(S)} y_S \leq w(v) \quad \forall v \in V \qquad (MWCDS - D) . \\
& y_S \geq 0
\end{aligned}
$$

y_S is the components of vector $\bar{y}$. The $MWCDS - D$ problem is to find a feasible vector $\bar{y}$ which maximizes the sum $F_{\bar{y}} \triangleq \sum_{S \subseteq V : S \neq \emptyset} f(S) \cdot y_S$. For more details about the $Primal - Dual$ relaxation, we will refer the reader to [7].

Let $\bar{x}$ be some feasible solution to $MWCDS - IP$, and $\bar{x}^*$ is the optimal solution. Consider that $\bar{x}_{LP}^*$ is the optimal solution to $MWCDS - LP$. Also, let $\bar{y}$ be some feasible dual solution to $MWCDS - D$, and $\bar{y}^*$ is the optimal solution. From the weak duality theorem we can state that

$$W_{\bar{x}} \geq W_{\bar{x}^*} \geq W_{\bar{x}_{LP}^*} = F_{\bar{y}^*} \geq F_{\bar{y}} \,. \tag{11}$$

Now, we argue that MWCDS and $MWCDS - IP$ are equivalent problems. Let $\bar{\sigma}$ be the equivalent solution of cds^* to $MWCDS - IP$, i.e. a binary vector with ones for each $v \in cds^*$. Symmetrically, let χ be the equivalent solution of $\bar{x}^*$ to MWCDS problem, i.e. the set of nodes with value 1 in $\bar{x}^*$. Note that $OPT_{cds^*} \triangleq Weight\,(cds^*) = W_{\bar{\sigma}}$ and $Weight\,(\chi) = W_{\bar{x}^*}$. We denote by $\mathscr{S}$ the collection of all subsets $\mathscr{S} \triangleq \{S \colon S \subseteq V, S \neq \emptyset\}$. We will show that $\bar{\sigma}$ is an optimal solution for $MWCDS - IP$.

Lemma 8. $\bar{\sigma}$ *is a feasible solution to the $MWCDS - IP$ problem.*

Proof. We have to show that $\bar{\sigma}$ complies with the constraints $\sum_{v \in N(S)} \sigma_v \geq f\,(S) \quad \forall S \in \mathscr{S}$ of the problem. In order to prove so, we partition the collection $\mathscr{S}$ into three possible cases. First, S contains cds^*; Second, S does not contain cds^* and $f\,(S) = 0$; And third, S does not contain cds^* and $f\,(S) = 1$. If S contains a DS, then also $f\,(S) = 0$. Hence, the first two cases are trivial. As for the third case, we claim that there is at least one node $v \in N\,(S)$ which $\sigma_v = 1$. Recall that $\sigma_v = 1$ if and only if $v \in cds^*$. If S and cds^* are disjoint sets, then cds^* dominates all S nodes. Therefore, each node $u \in S$ has a neighbor $v \in cds^*$ (i.e. $v \in N\,(S)$). If S and cds^* are not disjoint sets, then at least one node $u \in S \cap cds^*$ must have a neighbor $v \in cds^* \setminus S$ (i.e. $v \in N\,(S)$), since cds^* is connected. To conclude, $\bar{\sigma}$ is a binary vector that satisfies all the constraints, thus it is a feasible solution to $MWCDS - IP$. $\qquad\square$

Lemma 9. χ *is a feasible solution to the MWCDS problem.*

Proof. We first prove that χ is a DS. Since its equivalent, $\bar{x}^*$, is a solution to $MWCDS - IP$, it must comply with the constraints $\sum_{v \in N(S)} x_v^* \geq f\,(S) \quad \forall S \in \mathscr{S}$. Let us assume in contradiction that χ is not a DS. Then, according to the definition $f\,(\chi) = 1$. For all nodes $v \in V \setminus \chi$ holds $x_v^* = 0$, so $\sum_{v \in N(\chi)} x_v^* = 0$. This is in contradiction to the assumption that $\bar{x}^*$ satisfies the constraint $\sum_{v \in N(\chi)} x_v^* \geq f\,(\chi)$. Therefore, χ must be a DS.

Now we prove that χ is a CDS. Since its equivalent, $\bar{x}^*$, is an optimal solution to $MWCDS - IP$, it must comply with all the constraints $\sum_{v \in N(S)} x_v^* \geq f\,(S) \quad \forall S \in \mathscr{S}$ and has the minimum total weight. Let us assume in contradiction that χ is not connected. Hence, there are at least two maximal connected components in the induced subgraph of χ in G. Let us consider one of them as $C_\chi^1 = (V_C, E_C)$. In the case where $f\,(V_C) = 0$, V_C is a CDS, and therefore from Lemma 8 its equivalent vector satisfies all the constraints of $MWCDS - IP$. But, it has a smaller total weight than $\bar{x}^*$, and this is in contradiction to the minimality of $\bar{x}^*$. In the case where $f\,(V_C) = 1$, since C_χ^1 is maximal, then it must be $N\,(V_C) \cap \chi = \emptyset$. Therefore holds $\sum_{v \in N(V_C)} x_v^* = 0$. This is in contradiction to the assumption that $\bar{x}^*$ satisfies the constraint $\sum_{v \in N(V_C)} x_v^* \geq f\,(V_C)$. Hence χ must be connected. To conclude, χ is dominating and connected then it must be a CDS. $\qquad\square$

Theorem 10. *MWCDS and $MWCDS - IP$ are equivalent problems.*

Proof. From Lemma 8 $W_{\bar{\sigma}} \geq W_{\bar{x}^*}$ and from Lemma 9 $W_{\bar{\sigma}} = OPT_{cds^*} \leq Weight\,(\chi) = W_{\bar{x}^*}$, therefore must be hold that $OPT_{cds^*} = W_{\bar{\sigma}} = W_{\bar{x}^*}$. Since also $\bar{\sigma}$ is a feasible solution for $MWCDS - IP$, then the statement is true. $\qquad\square$

To conclude, let cds be some CDS in G. From Theorem 10 and Equation 11 we can state that:

$$Weight\,(cds) \geq OPT_{cds^*} = W_{\bar{x}^*} \geq F_{\bar{y}} \,. \tag{12}$$

So, if we find a CDS and a feasible dual solution, we can determine the approximation ratio between cds and cds^*.

Algorithm 1 Constructs an independent dominating set

Input: A connected UDG $G = (V, E)$ with $|V| \geq 2$, a node weight function $w(\cdot) : V \to R^+$

Ensure: An independent dominating set ds and a lower bound $LB1$

1: $ds \leftarrow \emptyset$, $A \leftarrow \emptyset$, $\mathscr{S} \leftarrow \{\{v\} : v \in V\}$, $c_v \triangleq \frac{1}{100} \cdot w(v)$

2: $y(\{v\}) \leftarrow 0 \, \forall v \in V$, $g(\{v\}) \leftarrow 1 \, \forall v \in V$

3: **repeat** // The algorithm steps

4:       Find node $v \in V$ that minimizes $\epsilon \leftarrow \epsilon(v)$

5:       $A \leftarrow A \cup \{v\}$

6:       **if** $v \notin N(ds) \cup ds$ **then** $ds \leftarrow ds \cup \{v\}$

7:       **for** each $S \in \mathscr{S}$ **do**

8:          $y(S) \leftarrow y(S) + g(S)\epsilon$

9:          **if** $v \in N(S)$ **then** $g(S) \leftarrow 0$

10:      **end for**

11:      **for** each $u \in V \setminus A$ **do**

12:         **if** $\sum_{S \in \mathscr{S} : u \in N(S)} g(S) = 0$ **then**

13:            $S' \leftarrow V \setminus \{u\}$

14:            $\mathscr{S} \leftarrow \mathscr{S} \cup \{S'\}$, $y(S') \leftarrow 0$, $g(S') \leftarrow 1$

15:         **end if**

16:      **end for**

17: **until** $(N(ds) \cup ds = V)$

18: **return** $\left\{ ds, \quad LB1 \leftarrow \sum_{v \in V} y(\{v\}) \right\}$

Definition 11 (Approximation ratio α). A feasible approximation ratio is based on a feasible CDS (cds) and a feasible dual solution ($\bar{y}$), and is defined as follows:

$$\alpha \triangleq \frac{Weight(cds)}{F_{\bar{y}}} \geq \frac{Weight(cds)}{OPT_{cds^*}} \,.$$

5 The Algorithm

5.1 Approximation algorithm

Here, we introduce our algorithm for constructing a data gathering tree which is based on the $Primal - Dual$ method, on Theorem 7 and on the conclusions of Subsection 4.2. The algorithm consists of three main stages (Algorithm 1, Algorithm2 and Algorithm 3). Algorithm 1 finds an independent DS (IDS), while constructing a feasible dual solution. Algorithm 2 creates a CDS based on the IDS from the previous stage, and update the dual solution. Finally, Algorithm 3 builds a gathering tree and finds an ideal MULE location. The BackBone of the gathering tree consists of the CDS from the previous stage.

Algorithm 1 Receives a connected UDG graph $G = (V, E)$ with at least two nodes, and a positive node weight function $w(\cdot) : V \to R^+$. The algorithm returns ds which is an IDS of nodes, and $LB1$ which is a lower bound value for the weight of the optimal solution to the MWCDS problem. The algorithm works in steps to find ds, while constructing a feasible dual solution which will serve as the lower bound.

Initially, ds is an empty set, the node capacity c_v is $\frac{1}{100} \cdot w(v)$, and $\mathscr{S}$ consists only of the $active$ subsets $\{\{v\} : v \in V\}$. An $active$ subset S is a subset of which the algorithm is willing to determine its y_S value for the dual solution. For all the other subsets in $\{S : S \subseteq V, S \neq \emptyset\}$, consider the value 0. $y(S)$ is the y_S value determined by the algorithm to subset S at the current step, and is initialized to 0. Node v is said to be "$packed$" if it satisfies the following equality: $\sum_{S \in \mathscr{S} : v \in N(S)} y(S) = c_v$. Subset S is said to be "$restricted$" if

it has a *packed* neighbor node, and therefore its $y(S)$ value can not be increased. $g(S)$ is an indicator with value 0 for *restricted* subsets. Formally, it is defined as follows:

$$g(S) = \begin{cases} 0 & if \ \exists v \in V : v \in N(S), \sum_{S' \in \mathscr{S} : v \in N(S')} y(S') = c_v \\ 1 & otherwise \end{cases}.$$

$g(S)$ is initialized to 1. The algorithm uses the potential function $\epsilon(\cdot)$ to select nodes, and this function is defined as follows:

$$\epsilon(v) = \begin{cases} \infty & for \ \sum_{S \in \mathscr{S} : \, v \in N(S)} g(S) = 0 \\ \frac{c_v - \sum_{S \in \mathscr{S} : \, v \in N(S)} y(S)}{\sum_{S \in \mathscr{S} : \, v \in N(S)} g(S)} & otherwise \end{cases}. \tag{13}$$

At each step, the algorithm selects the node with the minimum potential $\epsilon(\cdot)$ (line 4). Let us say node v. If the node is independent in ds then it is added to ds (line 6). The $y(S)$ value is uniformly increased by $\epsilon(v)$ for all *active* subsets that are not *restricted* (line 8). Later, we claim that the selected node is *packed* after the uniform increase of $y(S)$. Since node v is *packed*, the algorithm updates all subsets that are adjacent to v to be *restricted* (line 9). For each node that is not yet selected (not *packed*) and all subsets adjacent to it are *restricted*, the algorithm creates a new subset in $\mathscr{S}$, which is adjacent only to it (lines 11 - 16).

The algorithm ends once ds becomes a DS (line 17). The algorithm returns ds and the sum $\sum_{v \in V} y(\{v\})$ as $LB1$ (line 18).

Proposition 12. *Each node selected by Algorithm 1 is packed.*

Proof. The algorithm adds $\epsilon(v)$ to $y(S)$ of each *un-restricted* subset S that $v \in N(S)$. So we can write the following equation for the selected node v:

$$\sum_{S \in \mathscr{S} : \, v \in N(S)} (y(S) + g(S)\epsilon(v)) =$$

$$\sum_{S \in \mathscr{S} : \, v \in N(S)} y(S) + \frac{c_v - \sum_{S \in \mathscr{S} : \, v \in N(S)} y(S)}{\sum_{S \in \mathscr{S} : \, v \in N(S)} g(S)} \sum_{S \in \mathscr{S} : \, v \in N(S)} g(S) = c_v.$$

$\square$

Algorithm 2 Receives a connected UDG graph $G = (V, E)$ with at least two nodes, a positive node weight function $w(\cdot) : V \to R^+$, and ds which is an IDS in G. The algorithm returns cds which is a CDS of nodes, and $LB2$ which is a lower bound value on the weight of the optimal solution to the MWCDS problem. The algorithm works in steps to find cds, while constructing a feasible dual solution which will serve as the lower bound.

Initially, cds contains the ds nodes, the node capacity c_v is $\frac{99}{100} \cdot w(v)$, and $\mathscr{S}$ consists only of the *active* subsets $\{\{v\} : v \in ds\}$. $y(S)$ is initialized to $\frac{99}{500} \cdot (w(v) - 2)$ for each subset $\{v\} : v \in ds$, and $g(S)$ is initialized to 1.

At each step, the algorithm selects the node with the minimum potential $\epsilon(\cdot)$ (line 4). Let us consider node v. If the node is adjacent to at least two *un-restricted* subsets, then it is added to cds (line 8). For each subset adjacent to v that does not contain a node from $N(\{v\}) \cap cds$, the algorithm also adds to cds a neighbor from that subset (lines 10 - 11). See Figure 3 for illustration of adding nodes. Node v is selected, green nodes from ds, blue from $cds \setminus ds$, and reds are the other nodes that have been *packed*. In this situation, v and the red node are added to cds. Next, the $y(S)$ value is uniformly increased by $\epsilon(v)$ for all *active* subsets that are not *restricted* (line 15). We later claim that the selected node is *packed*, and therefore the algorithm updates all subsets that are adjacent to v, to be *restricted* (line 16). The algorithm adds a new subset to $\mathscr{S}$ containing v and all its adjacent subsets that are not *restricted*.

The algorithm ends once $\mathscr{S}$ contains only one subset that is not *restricted* (line 20). The algorithm returns cds and the sum $\sum_{v \in ds} y(\{v\})$ as $LB2$ (line 21).

Algorithm 2 Constructs a connected dominating set

Input: A connected UDG $G = (V, E)$ with $|V| \geq 2$, a node weight function $w(\cdot) : V \to R^+$, and an independent dominating set ds.

Ensure: A connected dominating set cds and a lower bound $LB2$.

1: $cds \leftarrow ds$, $\quad \mathscr{S} \leftarrow \{\{v\} : v \in ds\}$, $\quad c_v \triangleq \frac{99}{100} \cdot w(v)$

2: $y(\{v\}) \leftarrow \frac{99}{500} \cdot (w(v) - 2) \; \forall v \in ds$, $\quad g(\{v\}) \leftarrow 1 \; \forall v \in ds$

3: **repeat** // The algorithm steps

4: $\quad$ Find node $v \in V$ that minimizes $\epsilon \leftarrow \epsilon(v)$

5: $\quad \mathscr{S}^1 \leftarrow \{S \in \mathscr{S} : g(S) = 1, \, S \cap N(\{v\}) \neq \emptyset\}$

6: $\quad \mathscr{S}^2 \leftarrow \{S \in \mathscr{S} : g(S) = 1, \, S \cap N(\{v\}) \cap cds \neq \emptyset\}$

7: $\quad$ **if** $\sum_{S \in \mathscr{S} : v \in N(S)} g(S) > 1$ **then**

8: $\quad\quad cds \leftarrow cds \cup \{v\}$

9: $\quad\quad$ **if** $\mathscr{S}^1 \setminus \mathscr{S}^2 \neq \emptyset$ **then**

10: $\quad\quad\quad S \leftarrow \{u : u \text{ is one neighbor of } v \text{ from each } S' \in \mathscr{S}^1 \setminus \mathscr{S}^2\}$

11: $\quad\quad\quad cds \leftarrow cds \cup S$

12: $\quad\quad$ **end if**

13: $\quad$ **end if**

14: $\quad$ **for** each $S \in \mathscr{S}$ **do**

15: $\quad\quad y(S) \leftarrow y(S) + g(S) \epsilon$

16: $\quad\quad$ **if** $v \in N(S)$ **then** $g(S) \leftarrow 0$

17: $\quad$ **end for**

18: $\quad S \leftarrow \left(\bigcup_{S' \in \mathscr{S}^1} S' \right) \cup \{v\}$

19: $\quad \mathscr{S} \leftarrow \mathscr{S} \cup \{S\}$, $\quad y(S) \leftarrow 0$, $\quad g(S) \leftarrow 1$

20: **until** $\left(\sum_{S \in \mathscr{S}} g(S) = 1 \right)$

21: **return** $\left\{ cds, \quad LB2 \leftarrow \sum_{v \in ds} y(\{v\}) \right\}$
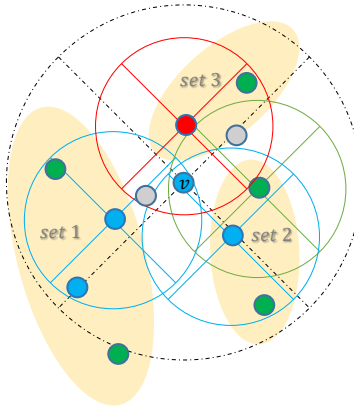

Figure 3: Adding nodes to cds in Algorithm. 2

Algorithm 3 Constructs a data gathering tree

Input: A connected UDG $G = (V, E)$ with $|V| \geq 2$

Ensure: A gathering tree $T = (V, E_T)$, the selected location for the MULE m, and the MWCDS approximation ratio α

1: $E_T \leftarrow \emptyset$, $cds \leftarrow \emptyset$
2: **for** each $m' \in V$ **do**
3:     $w(v) \leftarrow (2 \cdot dist\{m', v\} + C) \ \forall v \in V$
4:     $\{ds', LB1\} \leftarrow$ Algorithm 1 $(G, w(\cdot))$
5:     $\{cds', LB2\} \leftarrow$ Algorithm 2 $(G, w(\cdot), ds')$
6:     **if** $\sum_{v \in cds'} w(v) < \sum_{v \in cds} w(v)$ **then**
7:       $m \leftarrow m'$, $LB \leftarrow LB1 + LB2$, $cds \leftarrow cds'$
8:       $r \leftarrow$ The first node selected in ds' (at the first step of Algorithm 1)
9:     **end if**
10: **end for**
11: $SubG \leftarrow$ The induced subgraph of cds in G
12: $\bar{p} \leftarrow BFS(SubG, r)$ // Compute the predecessors of the nodes $v \in cds$ using BFS algorithm, starting from the root r
13: **for** each $v \in cds \setminus \{r\}$ **do**
14:     $E_T \leftarrow E_T \cup \{DirectedEdge(v \to p_v)\}$ // p_v is the predecessor of node v (a component of $\bar{p}$)
15: **end for**
16: **for** each $u \in V \setminus (cds \cup \{r\})$ **do**
17:     $v \leftarrow \arg\min_{v \in cds} \{dist\{u, v\}\}$
18:     $E_T \leftarrow E_T \cup \{DirectedEdge(u \to v)\}$
19: **end for**
20: **return** $\{T = (V, E_T) , \ \ m , \ \ \alpha \leftarrow \sum_{v \in cds} w(v)/LB\}$

Proposition 13. *Each node selected by Algorithm 2 is packed.*

Proof. Stems from the fact that the uniform increase of the $y(S)$ value is identical to Algorithm 1, and from Proposition 12. $\qquad\square$

Algorithm 3   Receives a connected UDG graph $G = (V, E)$ with at least two nodes. The algorithm returns a data gathering tree $T = (V, E_T)$ in graph G, the selected MULE location m, and the approximation ratio α (from Theorem 7).

     The algorithm searches iteratively for the minimal weight CDS produced by Algorithm 2, over all possible MULE locations. The node weight function is determined to be $w(v) = (2 \cdot dist\{m, v\} + C)$, according to Theorem 7. The algorithm uses Algorithm 1 to produce an IDS for Algorithm 2. The CDS with minimal weight is assigned to cds, the corresponding MULE location is assigned to m, LB is the corresponding sum $LB1 + LB2$, and r is the first node selected for the corresponding ds'.

     After selecting the minimal weight CDS, the algorithm uses the *BFS algorithm* to construct a directed spanning tree towards the root node r, in the induced subgraph of cds in G (lines 11 - 15). In this way, the cds nodes compose the BackBone of the gathering tree. Then, each node in $V \setminus cds$ is connected by an outgoing edge to the nearest node in cds.

     Finally, the algorithm returns the result tree as T, the MULE location m, and the approximation ratio α of the result.

5.2   Correctness

In this section, we will show the correctness of our algorithms. We will emphasize in advance that the algorithms are required to receive a connected UDG graph with at least two nodes in order to return correct

structures of IDS, CDS, and gathering tree. However, for the approximation ratio and for the lower bounds, additional constraints are required. For correct lower bounds ($LB1$ and $LB2$) and the α parameter, the graph should have a diameter of at least three. For correct approximation of the $(1,1) - MULE - UDG$ problem (i.e. $(1 + \varepsilon)$), it is required to be hold $\underset{v \in V_{BB}^*}{Average} \left[dist \left\{ m^*, v \right\} \right] > 1.3$, according to Theorem 7. As for graphs that do not meet only the first condition (on the diameter), these graphs are spread over a bounded area, and we will address this situation later in Subsection 5.3. We will show in a different approach that the result gathering tree still ensures the approximation required. As for graphs that do not meet the second condition, this issue still remains open.

Definition 14 (A feasible dual solution and lower bound). Consider the MWCDS problem and the dual problem $MWCDS - D$. Vector $\bar{y}$ will be called a *feasible dual solution*, if $\bar{y}$ is proven to be a solution to the $MWCDS - D$ problem and satisfies all its constraints. A *feasible lower bound* is a value that has been proven to have a *feasible dual solution* which is well defined, whose sum $\sum_{S \subseteq V : S \neq \emptyset} f(S) \cdot y_S$ is equal to this value. y_S are the components of $\bar{y}$, V is the set of the graph nodes and $f(S)$ is defined in Equation 10.

Correctness of Algorithm 1. Let $\mathscr{S}'$ be the collection of all *active* subsets that Algorithm 1 maintains in the current step (i.e. $\mathscr{S}' \subseteq \mathscr{S}$), and let $y'(S)$ be the value the algorithm assigns to subset S in the current step. We define the vector $\bar{y}'$ of size $|\mathscr{S}|$ (where y'_S are its components) as follows:

$$y'_S = \begin{cases} y'(S) & for \ |S| = 1 \\ 0 & for \ |S| > 1 \end{cases} \quad \forall S \in \mathscr{S} . \tag{14}$$

ϵ is the potential value of the selected node in the current step. A is the set of nodes that the algorithm has selected up to the current step. From Lemma 19, Algorithm 1 returns an IDS. From Lemma 20, if the graph diameter is at least three then it returns a feasible lower bound.

Proposition 15. *Once node v is selected by Algorithm 1 it becomes packed and $\epsilon(v) = \infty$, and it remains so until the end.*

Proof. From Proposition 12, after node v is chosen by the algorithm it is *packed*. According to lines 7 and 9, after node v is chosen, there is $\sum_{S \in \mathscr{S}' : v \in N(S)} g(S) = 0$, and therefore by Equation 13 $\epsilon(v) = \infty$. The only place that can change the packaging condition or the ∞ potential value is in line 14, since only here the algorithm performs $g(S') \leftarrow 1$. But this assignment is performed only for new subsets. For both conditions, subset S' will influence node v only if $v \in N(S')$. New subsets are of the form $V \setminus \{u\}$, and they have only one neighbor which is u. Since node v is already selected and added to A, it is not contained in the set $V \setminus A$. Hence, according to line 11, the subset $V \setminus \{v\}$ will not be added later in the algorithm. Therefore, new subsets will not influence node v, which will remain *packed* and $\epsilon(v) = \infty$. $\square$

Proposition 16. *In Algorithm 1, if $g(S) = 0$ for all $S \in \mathscr{S}'$, then there must be $A = V$.*

Proof. Let us assume in contradiction that $g(S) = 0 \ \forall S \in \mathscr{S}'$, but $A \neq V$. If so, there must be a node $u \in V \setminus A$, and from lines 12 - 14, there must be a subset $V \setminus \{u\} \in \mathscr{S}'$ which is adjacent to u. Since u is the only neighbor of $V \setminus \{u\}$, and u is not yet selected by the algorithm, then the *if* statement in line 9 never met, nor does the assignment $g(V \setminus \{u\}) \leftarrow 0$. This is in contradiction to the assumption $g(S) = 0 \ \forall S \in \mathscr{S}'$, and therefore must be $A = V$. $\square$

Proposition 17. *The set ds constructed by Algorithm 1 is an IS, and if $A = V$ then ds must be a MIS.*

Proof. The set ds is IS, since, in line 6 a node is added to ds only if it is independent of the ds nodes. Note that by line 5, if there is a step where $A = V$, it requires that each node was selected at least once. Let us assume in contradiction that $A = V$, but the set ds is not a MIS. Then there is at least one node $v \in V \setminus ds$ that can be added to ds, so it must be $v \notin N(ds) \cup ds$. But since v is already chosen by the algorithm, it is in contradiction to the condition in line 6. Hence, ds must be MIS. $\square$

Proposition 18. *Each node is selected at most once during Algorithm 1.*

Proof. According to Proposition 15, the potential value of each node $v \in V$ selected by the algorithm becomes $\epsilon(v) = \infty$, and remains so until the end. We want to show that there is no step in which a node is selected with a potential value of $\epsilon(v) = \infty$. To do this, let us assume in contradiction that there is a step where node v is selected with $\epsilon(v) = \infty$, but the stop condition (line 17) has not yet met. Then, according to line 4, for each node $u \in V$ holds $\epsilon(u) = \infty$. Since $c_u < \infty$, then it must be the first case in Equation 13. The graph is connected with at least two nodes, so each subset in $\mathscr{S}'$ is adjacent to at least one node. Therefore, must be $g(S) = 0$ for all $S \in \mathscr{S}'$, and according to Proposition 16, must also hold $A = V$. Consequently, from Proposition 17, the set ds must be a MIS, hence it is also a DS. This is in contradiction to the assumption that $N(ds) \cup ds \neq V$. Therefore, the claim is correct and node can not be selected twice. $\qquad\square$

Lemma 19. *Given a connected UDG with at least two nodes and node weights $w(\cdot) : V \to R^+$, Algorithm 1 returns an IDS.*

Proof. First, we argue that the set ds contains at least one node. Since the graph is connected and has at least two nodes, and $\mathscr{S}'$ is initialized to be $\{\{v\} : v \in V\}$, then at the first iteration, each node $v \in V$ has at least one subset $S \in \mathscr{S}'$ for which $v \in N(S)$. Without loss of generality, let us consider node v and some subset S, where $v \in N(S)$. From line 2, $g(S) \leftarrow 1$, so for node v holds: $\sum_{S \in \mathscr{S}' : v \in N(S)} g(S) > 0$. Since also $c_v = \frac{1}{100} \cdot w(v) < \infty$, then from Equation 13, at least node v will be selected at the first iteration, and $\epsilon \leq \epsilon(v) < \infty$.

From line 17, we can see that the algorithm halts once set ds becomes a DS. In Proposition 18, we saw that each node can be selected at most once. Then the iterations must be halt, and this is as a result of the fact that the set ds has become a DS.

Combine all that with Proposition 17, we get that ds is both IS and DS. Therefore it is an IDS, and hence the lemma is correct. $\qquad\square$

Lemma 20. *Given a connected UDG with a diameter of at least three and node weights $w(\cdot) : V \to R^+$, Algorithm 1 returns a feasible lower bound, having $\bar{y}'$ as its feasible dual solution that also satisfies $\sum_{S \in \mathscr{S} : v \in N(S)} y'_S \leq \frac{1}{100} \cdot w(v) \ \forall v \in V$.*

Proof. We will prove by induction on the algorithm steps that it maintains the constraints $\sum_{S \in \mathscr{S}' : v \in N(S)} y'(S) \leq c_v \ \forall v \in V$. Since at the initialization phase $y'(S) = 0 \quad \forall S \in \mathscr{S}'$ (line 2) and since $c_v > 0$, then it is easy to see that the constraints are maintained for base case $step = 0$. The induction hypothesis is that the statement is correct up to $step = k$, and we will prove that the statement remains correct for $step = k+1$. Note, the only line that can violate the constraints is line 8, where $y'(S)$ is updated. In step $k + 1$, the algorithm selects node v. Therefore, according to Proposition 12, node v is *packed*, i.e. $\sum_{S \in \mathscr{S}' : v \in N(S)} y'(S) = c_v$, and according to Proposition 15, it remains *packed* until the end of the algorithm. Then, nodes where the statement can be violated are only those that have not yet been selected (i.e. from $V \setminus A$). Let us assume in contradiction that there is a node $u \in V \setminus A$ that after step $k + 1$ violates the statement, and let $y'_k(S)$ be the value of $y'(S)$ at step k. We can write:

$$
\begin{aligned}
\sum_{S \in \mathscr{S}' : u \in N(S)} y'(S) &> c_u \\
\sum_{S \in \mathscr{S}' : u \in N(S)} \left(y'_k(S) + g(S)\epsilon(v) \right) &> c_u \\
\epsilon(v) &> \frac{c_u - \sum_{S \in \mathscr{S}' : u \in N(S)} y'_k(S)}{\sum_{S \in \mathscr{S}' : u \in N(S)} g(S)} = \epsilon(u) .
\end{aligned}
$$

The result is in contradiction to the minimality of $\epsilon(v)$. So, the statement is correct for step $k + 1$. Hence, the algorithm satisfies the constraints.

$\mathscr{S}'$ contains subsets $\{v\} : v \in V$ (line 1) and $y'_S = 0 \ \forall S : |S| > 1$ (Equation 14), then we get $\sum_{S \in \mathscr{S} : v \in N(S)} y'_S = \sum_{S \in \mathscr{S}' : v \in N(S)} y'_S$. Since $\sum_{S \in \mathscr{S}' : v \in N(S)} y'(S) \leq c_v$, then $\epsilon(v) \geq 0$ (Equation 13). Combine it with line 8, we get $y'(S) \geq 0$. So, from Equation 14, $y'_S \leq y'(S)$. Recall that $c_v \triangleq \frac{1}{100} \cdot w(v)$. Then

15

the following holds:

$$\sum_{S \in \mathscr{S}\,:\,v \in N(S)} y'_S \leq \sum_{S \in \mathscr{S}'\,:\,v \in N(S)} y'(S) \leq c_v = \frac{1}{100} \cdot w(v) \leq w(v) \ \forall v \in V\,.$$

Since also $y'_S \geq 0$ (Equation 14), then vector $\bar{y}'$ satisfies all the constraints of $MWCDS - D$. Hence, $\bar{y}'$ is a feasible dual solution.

Since the diameter of the graph is at least three, no subset of a single node is a DS. That is, $f(\{v\}) = 1 \ \forall v \in V$ (Equation 10). Combine it with Equation 14 and line 18, we get

$$\sum_{S \subseteq V : S \neq \emptyset} f(S) \cdot y'_S = \sum_{S \subseteq V : S \neq \emptyset} y'_S = \sum_{S \subseteq V\,:\,|S|=1} y'(S) = \sum_{v \in V} y'(\{v\}) = LB1\,. \tag{15}$$

Therefore, by Definition 14, $LB1$ is a feasible lower bound and $\bar{y}'$ is its feasible dual solution. Hence, the lemma is correct. $\qquad \square$

Correctness of Algorithm 2. Let $\mathscr{S}''$ be the collection of the *active* subsets that Algorithm 2 maintains in the current step (i.e. $\mathscr{S}'' \subseteq \mathscr{S}$), and let $y''(S)$ be the value that the algorithm assigns to subset S in the current step. We define the vector $\bar{y}''$ of size $|\mathscr{S}|$ (where y''_S are its components) as follows:

$$y''_S = \begin{cases} y''(S) & for\ S = \{v\} : v \in ds \\ 0 & otherwise \end{cases} \quad \forall S \in \mathscr{S}\,. \tag{16}$$

ϵ is the potential value of the selected node in the current step. Let $\mathscr{S}^1$ be the collection of all *un-restricted* *active* subsets that contain some neighbor of the selected node, and $\mathscr{S}^2$ be the collection of all subsets in $\mathscr{S}^1$ that also contain a node which is both a neighbor of the selected node and belongs to the *cds* set. From Lemma 27, Algorithm 2 returns a CDS. From Lemma 28, if the graph diameter is at least three and node weights are $w(v) = 2 \cdot dist\{m, v\} + C$, then it returns a feasible lower bound.

Proposition 21. *At each step of Algorithm 2, if node v is contained in some active subset, then there must be an un-restricted subset which contains v.*

Proof. We will prove by induction on the steps, for each node separately. Without loss of generality, let v be the node. The base case is the earliest step in which v is contained in a subset. Let k_0 be this step and S_0 be this subset. If $v \in ds$, then by lines 1 and 2, the base case is trivial. If $v \notin ds$, then by line 18, v must be selected by the algorithm to be contained in S_0. By line 19, the condition holds for the base case. If v is never selected, then it is irrelevant. The induction hypothesis is that the statement holds for step $k > k_0$, and we will prove that the statement still holds for step $k + 1$. According to the induction hypothesis, at step k there is an *un-restricted* subset S' which contains v. Let u be the node selected at step $k + 1$. If $u \notin N(S')$, then by line 16, the situation remains the same. Let S be the new subset at step $k + 1$. If $u \in N(S')$, then $S' \in \mathscr{S}^1$ (line 5) and therefore $v \in S$ (line 18). In line 19, $g(S) \leftarrow 1$, so the statement holds. Hence the claim is correct. $\qquad \square$

Proposition 22. *After node v is selected by Algorithm 2 it becomes packed and $\epsilon(v) = \infty$. Once a node becomes packed or the potential value becomes ∞, it remains so until the end.*

Proof. From Proposition 13, after node v is selected by the algorithm it is *packed*. According to lines 14 and 16, after node v is selected, there is $\sum_{S \in \mathscr{S}''\,:\,v \in N(S)} g(S) = 0$, and therefore by Equation 13, $\epsilon(v) = \infty$. The only place that can change the *packing* condition or the ∞ potential value is in line 19, since only here the algorithm performs $g(S) \leftarrow 1$. But this assignment is performed only for new subsets. For both conditions, subset S will influence node v only if $v \in N(S)$ and the assignment $g(S) \leftarrow 1$ will be performed. Let us assume in contradiction that there is a new subset that influences v at some later step. Let k be the earliest step in which v is influenced. Let S be the new subset created in this step, and u be the selected node. S

16

consists of all nodes of subsets $S' \in \mathscr{S}'' : g(S') = 1, S' \cap N(\{u\}) \neq \emptyset$ and node u. None of these subsets can be adjacent to v at step $k - 1$, since $\epsilon(v) = \infty$. Therefore, the only possibility to influence v is if u and v are neighbors. From Proposition 21, there is a subset $S'' \in \mathscr{S}'' : v \in S'', g(S'') = 1$. By line 5, $S'' \in \mathscr{S}^1$, and therefore $v \in S$. This is in contradiction to the assumption that S influences v. Hence the claim is correct. $\qquad\square$

Proposition 23. *Algorithm 2 does not select a node with potential value ∞.*

Proof. Let us assume in contradiction that there is a step in which the algorithm selects a node with potential value ∞. Consider k to be the earliest step at which such a node is selected, and let v be this node. By line 4, must hold $\epsilon(u) = \infty$ for all $u \in V$. From Equation 13, and since $c_v < \infty$, it requires that $\sum_{S \in \mathscr{S}'' : u \in N(S)} g(S) = 0$ for all $u \in V$. Since the graph is connected, it implies that there is no *un-restricted* subset $S \in \mathscr{S}'' : |S| < |V|$. According to line 19, we know that there must be an *un-restricted* subset $S' \in \mathscr{S}''$. Then this subset must be $S' = V$. So it also has to be the only *un-restricted* subset. This is in contradiction to the stop condition (line 20) at the end of step $k - 1$. Hence the claim is correct. $\qquad\square$

Proposition 24. *At each step of Algorithm 2, the union of all un-restricted active subsets contains the set cds.*

Proof. By line 1, nodes from ds are contained in some *active* subset S. As for nodes from $cds \setminus ds$, each node had to be selected by the algorithm to be added to cds. Thus, by line 18, this node was also added to some *active* subset. According to Proposition 21, a node that is contained in some *active* subset must also be contained in an *un-restricted* subset. Hence, all the nodes in cds are included in the union. $\qquad\square$

Proposition 25. *During Algorithm 2, no node from subset ds will be selected.*

Proof. Let v be some node from ds. Since ds is an IS, at the initialization stage (line 1) there is no subset $S \in \mathscr{S}''$ adjacent to v. So, $\sum_{S \in \mathscr{S}'' : v \in N(S)} g(S) = 0$, and as a result $\epsilon(v) = \infty$. From propositions 22 and 23, the potential value of v remains ∞ until the end, and therefore the node will not be selected by the algorithm. $\qquad\square$

Proposition 26. *For each node $v \in S : S \in \mathscr{S}''$, if $v \notin cds$ then it must have a neighbor $u \in S \cap cds$.*

Proof. Consider some node v and some subset S, where $S \in \mathscr{S}''$ and $v \in S \setminus cds$. Of course $v \notin ds$, consequently v was not added to S at the initialization stage (step 0), but rather in a later step, say step $k > 0$. Note that from propositions 13, 22 and 23, after node v is selected, $\epsilon(v) = \infty$ until the end of the algorithm, and so it will not be selected again. Without loss of generality, let S be the first subset v added to. Otherwise, since v will not be selected again, by line 18 all other subsets containing v also contain all S nodes. Since ds is a DS and $v \notin ds$, then there is a node $u \in ds \cap N(\{v\})$. Note that in step 0, u was contained in some subset, and therefore, according to Proposition 21, in step $k - 1$, there is a subset $S' \in \mathscr{S}'' : u \in S', g(S') = 1$. In step k, S' will be included in the collection $\mathscr{S}^1$. We assumed that $v \notin cds$, so it must be because the condition in line 7 is not met. Thus it requires that $|\mathscr{S}^1| = 1$, and only S' contained in $\mathscr{S}^1$. Recall that the neighbor $u \in S' \cap ds$, hence $u \in S' \cap cds$. According to line 18, S contains v and all S' nodes, and therefore the claim is correct. $\qquad\square$

Lemma 27. *Given a connected UDG with at least two nodes, node weights $w(\cdot) : V \to R^+$, and an IDS, Algorithm 2 returns a CDS.*

Proof. First, we will show that the algorithm halts. From Proposition 13, the selected node is packed, and according to Proposition 22, it remains so until the end. Thus after a node is selected, its potential value will remain ∞. According to Proposition 23, this node will not be selected again. Hence, the algorithm must halts after at most $|V|$ steps.

We will show that the resulting cds is indeed CDS. Since cds contains ds then it is dominates all V nodes.

After the algorithm halts, by line 20, there must be $\sum_{S \in \mathscr{S}''} g(S) = 1$. According to Proposition 24, the only *un-restricted* subset $S \in \mathscr{S}''$ must contain cds. We will prove by induction on the algorithm steps that

17

it maintains the intersection $S' \cap cds$ connected for each $S' \in \mathscr{S}''$ (and especially for S).

According to line 1, initially, each subset $S' \in \mathscr{S}''$ contains a single node from ds. So the base case is trivial. The induction hypothesis is that the statement holds for step k, and we will prove that the statement still holds for step $k + 1$. The only new subset added during step $k + 1$ is in line 19. Consider S' to be the new subset and v to be the selected node. ds is a DS, and by Proposition 25, we know that v is not in ds. Then, there is a node $u \in ds$ which is $u \in N(\{v\})$. By line 1, u is contained in some subset, so according to Proposition 21, there is a subset $S'' : u \in S''$ that satisfies $g(S'') = 1$. Therefore, $\mathscr{S}^1$ contains at least the subset S''.

If the condition in line 7 is not met, then v is not added to cds, and $\mathscr{S}^1$ contains only S''. From the induction hypothesis the statement holds, since $S' \cap cds = (S'' \cup \{v\}) \cap cds = S'' \cap cds$.

If the condition in line 7 is met, then v is added to cds and $|\mathscr{S}^1| > 1$. We will show that $(S'' \cup \{v\}) \cap cds$ is CS, for all $S'' \in \mathscr{S}^1$. Therefore, by line 18, $S' \cap cds$ must also be connected. Note that $\mathscr{S}^2 \subseteq \mathscr{S}^1$. If $S'' \in \mathscr{S}^2$, then v has a neighbor $u \in S'' \cap cds$. Hence $(S'' \cup \{v\}) \cap cds$ is also CS. If $S'' \in \mathscr{S}^1 \setminus \mathscr{S}^2$, then by lines 9 - 12, the algorithm adds to cds some neighbor of v. Let u be this neighbor, and note that $u \notin cds$. From Proposition 26, u has a neighbor in $S'' \cap cds$, then $S'' \cap (cds \cup \{u\})$ is CS. After line 11, $S'' \cap cds$ remains connected, and so is $(S'' \cup \{v\}) \cap cds$.

The statement holds for step $k + 1$. Therefore, $S' \cap cds$ is a CS for each $S' \in \mathscr{S}''$.

If we consider the last step, $S \cap cds$ is also a CS. Since $S \cap cds = cds$, then cds must also be a CDS. Therefore, the lemma is correct. $\qquad\square$

Lemma 28. *Given a connected UDG with a diameter of at least three and node weights $w(v) = 2 \cdot dist\{m, v\} + C$, Algorithm 2 returns a feasible lower bound, having $\bar{y}''$ as its feasible dual solution that also satisfies $\sum_{S \in \mathscr{S} : v \in N(S)} y''_S \leq \frac{99}{100} \cdot w(v) \ \forall v \in V$.*

Proof. We will prove by induction on the algorithm steps that it maintains the constraints $\sum_{S \in \mathscr{S}'' : v \in N(S)} y''(S) \leq c_v \ \forall v \in V$. In the initialization phase $\mathscr{S}'' \triangleq \{\{v\} : v \in ds\}$, so $\sum_{S \in \mathscr{S}'' : v \in N(S)} y''(S) = \sum_{u \in ds : v \in N(\{u\})} y''(\{u\})$. Recall that $y''(\{u\})$ are initialized to $\frac{99}{500} \cdot (w(u) - 2)$. Since node weights are $w(u) \triangleq 2 \cdot dist\{m, u\} + C$ and the distance between a pair of neighbors in UDG is at most 1, then $w(u) \leq w(v) + 2 : v \in N(\{u\})$. ds is an IS in UDG. Therefore, each node in the graph can have at most 5 neighbors in ds (the kissing number problem). Formally, $|ds \cap N(\{v\})| \leq 5 \ \forall v \in V$. When we put it all together:

$$
\begin{aligned}
\sum_{S \in \mathscr{S}'' : v \in N(S)} y''(S) &= \sum_{u \in ds : v \in N(\{u\})} y''(\{u\}) \\
&= \sum_{u \in ds : v \in N(\{u\})} \tfrac{99}{500} \cdot (w(u) - 2) \\
&\leq \sum_{u \in ds : v \in N(\{u\})} \tfrac{99}{500} \cdot ((w(v) + 2) - 2) \\
&= |ds \cap N(\{v\})| \cdot \tfrac{99}{500} \cdot w(v) \\
&\leq 5 \cdot \tfrac{99}{500} \cdot w(v) = \tfrac{99}{100} \cdot w(v) = c_v .
\end{aligned}
$$

Thus the statement is correct for base case $step = 0$. The induction hypothesis is that the statement is correct up to $step = k$, and we will prove that the statement remains correct for $step = k + 1$. Note, the only line that can violate the constraints is line 15, where $y''(S)$ is updated. In step $k + 1$, the algorithm selects node v. Therefore, according to Proposition 13, node v is *packed*, i.e. $\sum_{S \in \mathscr{S}'' : v \in N(S)} y''(S) = c_v$, and according to Proposition 22, it remains *packed* until the end of the algorithm. Then, nodes where the statement can be violated are only those that have not yet been selected. Let us assume in contradiction that there is a node u that has not yet been *packed* and after step $k + 1$ violates the statement. Let $y''_k(S)$

be the value of $y''(S)$ at step k. So, we can write:

$$\sum_{S \in \mathscr{S}'' : \, u \in N(S)} y''(S) > c_u$$
$$\sum_{S \in \mathscr{S}'' : \, u \in N(S)} \left(y_k''(S) + g(S)\,\epsilon(v)\right) > c_u$$
$$\epsilon(v) > \frac{c_u - \sum_{S \in \mathscr{S}'' : \, u \in N(S)} y_k''(S)}{\sum_{S \in \mathscr{S}'' : \, u \in N(S)} g(S)} = \epsilon(u) \; .$$

In contradiction to the minimality of $\epsilon(v)$. Thus the statement is correct for step $k+1$. Hence, the algorithm satisfies the constraints.

$\mathscr{S}''$ contains subsets $\{v\} : v \in ds$ (line 1) and $y_S'' = 0 \; \forall S \notin \{\{v\} : v \in ds\}$ (Equation 16), then we get $\sum_{S \in \mathscr{S} : \, v \in N(S)} y_S'' = \sum_{S \in \mathscr{S}'' : \, v \in N(S)} y_S''$. Since $\sum_{S \in \mathscr{S}'' : \, v \in N(S)} y''(S) \leq c_v$, then $\epsilon(v) \geq 0$ (Equation 13). By combining this with line 15, we get $y''(S) \geq 0$. Consequently, from Equation 16, $y_S'' \leq y''(S)$. Recall that $c_v \triangleq \frac{99}{100} \cdot w(v)$. Then the following holds:

$$\sum_{S \in \mathscr{S} : \, v \in N(S)} y_S'' \leq \sum_{S \in \mathscr{S}'' : \, v \in N(S)} y''(S) \leq c_v = \frac{99}{100} \cdot w(v) \leq w(v) \; \forall v \in V \; .$$

Since also $y_S'' \geq 0$ (Equation 16), then vector $\bar{y}''$ satisfies all the constraints of $MWCDS - D$. Hence, $\bar{y}''$ is a feasible dual solution.

Since the diameter of the graph is at least three, so no subset of a single node is a DS. That is, $f(\{v\}) = 1 \; \forall v \in ds$ (Equation 10). Combine it with Equation 16 and line 21, we get

$$\sum_{S \subseteq V : S \neq \emptyset} f(S) \cdot y_S'' = \sum_{S \in \mathscr{S}} y_S'' = \sum_{v \in ds} y''(\{v\}) = LB2 \; . \tag{17}$$

Therefore, by Definition 14, $LB2$ is a feasible lower bound and $\bar{y}''$ is its feasible dual solution. Hence, the lemma is correct. $\qquad\square$

Correctness of Algorithm 3. From Lemma 29, Algorithm 3 returns a data gathering tree and its corresponding MULE location. From Lemma 30, if the graph diameter is at least three then it returns a feasible approximation ratio.

Lemma 29. *Given a connected UDG with at least two nodes, Algorithm 3 returns a data gathering tree and its corresponding MULE location.*

Proof. According to lines 14 and 18, the structure consists of directed edges.

Since the node weight function is $0 < w(\cdot) < \infty$ (line 3) and $|V| \geq 2$, then according to Lemma 19 the subset ds' that produces Algorithm 1 (line 4) is an IDS, and according to Lemma 27, the subset cds' produced by Algorithm 2 (line 5) is a CDS. From line 7, also cds is a CDS. Let $SubG$ be the induced sub graph of cds in G. So, since G is a connected graph and cds is a CS, then $SubG$ is a connected graph. By line 8, $r \in ds'$, therefore, r is a node in $SubG$. In line 12, since the BFS algorithm starts at node r and the graph is connected, then at line 14 at least one incoming edge must be added to r. From lines 13, 14, 16 and 18, we see that each node (except r) has a directed outgoing edge in the structure. Hence, we can say that the structure spans all the nodes in G.

From the BFS mechanism to finding shortest paths, and since for each node in $cds \setminus \{r\}$, an added edge is directed to its predecessor, after the for loop of line 13, r is reachable from all $cds \setminus \{r\}$ nodes in the sub graph $SubG$. After the for loop of line 16, it can be reached from any of the other nodes of V to some node in cds. Therefore, r is also reachable from all the nodes in the structure.

Since G is a connected UDG and cds is a DS, note that each directed edge in the structure is added only between a pair of neighbors, so it is also an undirected edge in G.

We have seen that the structure that constructs Algorithm 3 is a directed spanning tree in G, and that r is reachable from all nodes. Thus, by Definition 1, it can be said that the structure is a data gathering tree and r is its root.

According to the last iteration where line 7 is performed, we can see that m is the MULE location used for building cds, which is used later on to build the gathering tree. Hence, the lemma is correct. $\qquad\square$

Lemma 30. *Given a connected UDG with a diameter of at least three, Algorithm 3 returns a feasible approximation ratio α.*

Proof. Consider $\bar{y}'$ and $\bar{y}''$, to be the vector solution that constructs Algorithm 1 (Equation 14) and Algorithm 2 (Equation 16), respectively. According to lemmas 20 and 28, since node weights are $w\,(v) \triangleq 2 \cdot dist\,\{m, v\} + C$ and the graph diameter is at least three, then holds $\sum_{S \in \mathscr{S} \,:\, v \in N(S)} y'_S \leq \frac{1}{100} \cdot w\,(v)$ and $\sum_{S \in \mathscr{S} \,:\, v \in N(S)} y''_S \leq \frac{99}{100} \cdot w\,(v)$ for all $v \in V$. Consider vector $\bar{y}$ (where y_S are its components) which is defined as follows: $\bar{y} \triangleq \bar{y}' + \bar{y}''$. So we can write:

$$
\begin{aligned}
\sum_{S \in \mathscr{S} \,:\, v \in N(S)} y_S &= \sum_{S \in \mathscr{S} \,:\, v \in N(S)} (\bar{y}'_S + \bar{y}''_S) \\
&\leq \tfrac{1}{100} \cdot w\,(v) + \tfrac{99}{100} \cdot w\,(v) \\
&= w\,(v) \ \ \forall v \in V \,.
\end{aligned}
$$

Since also $y_S = \bar{y}'_S + \bar{y}''_S \geq 0$, then vector $\bar{y}$ satisfies all the constraints of $MWCDS - D$. Hence, $\bar{y}$ is a feasible dual solution. Using equations 15 and 17, we can also write:

$$
\sum_{S \subseteq V : S \neq \emptyset} f\,(S) \cdot y_S = \sum_{S \subseteq V : S \neq \emptyset} f\,(S) \cdot y'_S + \sum_{S \subseteq V : S \neq \emptyset} f\,(S) \cdot y''_S = LB1 + LB2 \,.
$$

Therefore, by Definition 14, $LB = LB1 + LB2$ is a feasible lower bound.

In Lemma 29, we showed that cds is a CDS. Then by line 20 and from Definition 11, we can say that the value α returned by algorithm 3 is a feasible approximation ratio. $\qquad\square$

5.3 Performance guarantees

In this section we show performance guarantees of Algorithm 3 (including algorithms 1 and 2) in the context of approximation and runtime. We analyze the approximation ratio of the gathering tree produced by Algorithm 3 with respect to the optimal gathering tree for the $(1, 1) - MULE - UDG$ problem. According to Theorem 7, we show that if the input graph satisfies the condition $\underset{v \in V^*_{BB}}{Average}\,[dist\,\{m^*, v\}] > 1.3$, then the algorithm achieves an approximation of $20 + \varepsilon$. It should be noted that for graphs that do not meet this condition, the problem still remains open. From now on, let us consider only graphs that satisfy the condition. Lemma 32 and Lemma 33 prove the approximation ratio. Further, we analyze the algorithm runtime and present optimizations that can achieve time complexity of $O\left(n^3 \cdot \Delta\,(G)\right)$, where $\Delta\,(G)$ is the maximum degree in the graph and n is the number of nodes.

Algorithm 3 produces a $(20 + \varepsilon) - approximate$ solution. Let us refer to nodes in $cds \setminus ds$ as the "connectors", since they connect the ds nodes.

Lemma 31. *There is an injective mapping function from each connector to a single node in ds, such that no more than two connectors are mapped to one node. The connector is at most one unit away from the node, and if there is an additional connector, then it is at most two units away.*

Proof. In order to prove this claim, we first build a spanning tree $T' = (cds, E')$, which spans all cds nodes. Then, based on T', we explain the mapping of each *connector*.

The construction of T' is based on the decisions made by Algorithm 2. For each node v selected at each step, we add two types of edges. The first type is for each subset $S \in \mathscr{S}^2$, we add the edge (v, u) to E', where $u \in S \cap N\,(\{v\}) \cap ds$ if there is one, and if not, $u \in S \cap N\,(\{v\}) \cap cds$. The second type of edges is for each subset $S \in \mathscr{S}^1 \setminus \mathscr{S}^2$, we add the edge (v, u) to E', where u is a neighbor according to the choice of the algorithm in line 10. At the end, we go over all edges in E' and delete edges with nodes in $V \setminus cds$.

The proof that $T' = (cds, E')$ is a spanning tree for cds is done by induction on the algorithm steps. Recall that $\mathscr{S}''$ is the collection of all *active* subsets of Algorithm 2. We want to show that at each step and for each *un-restricted active* subset $S \in \mathscr{S}''$, the induced subgraph of $S \cap cds$ in T' is a spanning tree for the nodes in $S \cap cds$. The base case, step 0, is trivial. The induction hypothesis is that the statement

holds for step $k > 0$, and we will prove that the statement still holds for step $k + 1$. Consider S' to be the new subset created in step $k + 1$. S' contains all subsets of $\mathscr{S}^1$. From the induction hypothesis for step k and since each $S'' \in \mathscr{S}^1$ is *un-restricted*, the induced subgraph of $S'' \cap cds$ in T' is a spanning tree for $S'' \cap cds$. Let v be the selected node. Since ds is a DS, then v has a neighbor in ds. At initialization, this neighbor is contained in some subset. So, according to Proposition 21, implies that $\left|\mathscr{S}^1\right| > 0$. If at the end of the algorithm $v \in cds$, then for each $S'' \in \mathscr{S}^1$ one edge is added to E', and connects $S'' \cap cds$ (also its sub spanning tree) to v. If eventually $v \notin cds$, then $v \notin S'' \cap cds$ and its edges are deleted at the end. Since all edges are added in a star form around v, then the result structure is still a tree, and it also spans node v. Hence, it is a spanning tree for $S' \cap cds$. According to the induction principle, the statement is correct for every step. In particular for the last step, in which $\mathscr{S}''$ contains only one *un-restricted* subset S. From Proposition 24 we know that $cds \subseteq S$, so T' is a spanning tree for cds.

The mapping function between the *connectors* and the ds nodes is described as follows. Since the algorithm adds to cds, only *connectors* with at least two edges in E', then the *connectors* can not be leaves in T'. Consider the root node r, we know that $r \in ds$. We go over all edges of T', and each edge we directed towards r. Each *connector* is mapped to the nearest (by hops) node in ds which has a directed path to that *connector*. In case of multiple choice, select one arbitrarily.

The proof that we do not map to one node in ds, more than two *connectors* that meet the distance requirements is as follows. We will show that each *connector* $v \in cds \setminus ds$ has a node $u \in ds$ so that there is a directed path from u to v of a length no more than two hops. If v has an incoming edge from a node in ds, then we are done. Otherwise, let us consider v' to be one of the *connectors* of the incoming edges (note, since v is not a leaf then there must be one). Since T' is a directed tree towards a particular node, then v' has only one outgoing edge, and this edge is to v. All other edges of v' must be incoming. We have already seen that $\left|\mathscr{S}^1\right| > 0$ since each selected node has a neighbor from ds in G, so each *connector* has a neighbor from ds, also in tree T'. Then, one of the incoming edges of v' must be from a node in ds, and let us denote this node by u. Thus, we have the path $u \to v' \to v$ in T', and we are done.

Since T' is a directed tree towards a particular node, each node has only one outgoing edge. Therefore, each node from ds can reach at most two *connectors* at a distance of two hops in T'. These are the only two *connectors* that can be mapped to this node. Since we are dealing with an UDG, the first must be at most one unit away, and the second at most two units. Hence, we complete the proof. $\qquad\square$

Lemma 32. *Given a connected UDG with **a diameter of at least three**, Algorithm 3 returns a* $(20 + \varepsilon) - approximate$ *solution to the* $(1,1) - MULE - UDG$ *problem.*

Proof. Recall that $w(v) = 2 \cdot dist\{m, v\} + C$. According to Lemma 31, each node v from ds has at most two *connectors* in $cds \setminus ds$, where one *connector* has a weight greater by 2 and the other by 4, than v's weight. Therefore, we can bound cds weight by $Weight(cds) \leq \sum_{v \in ds} (3 \cdot w(v) + 6)$. In Algorithm 2 line 2, we assign initial y's values, then we can bound $F_{\bar{y}}$ from below by $F_{\bar{y}} \geq \sum_{v \in ds} \frac{99}{500} \cdot (w(v) - 2)$. Since both sums contain only the ds nodes, we can calculate the ratio even for just a single node. Note that $w(v) \geq C$, and since $0 < R_M < 0.3$, then $C > 14.5$ (Figure 2). So from Definition 11, we can state that the approximation ratio α is:

$$\frac{Weight(cds)}{F_{\bar{y}}} \leq \frac{\sum_{v \in ds} (3 \cdot w(v) + 6)}{\sum_{v \in ds} \frac{99}{500} \cdot (w(v) - 2)} = \frac{3 \cdot w(v) + 6}{99/500 \cdot (w(v) - 2)} =$$

$$= \frac{3}{99/500} + \frac{6 + 6}{99/500 \cdot (w(v) - 2)} \leq \frac{500}{99}\left(3 + \frac{12}{(14.5 - 2)}\right) = 20 . \tag{18}$$

From Equation 12, $OPT_{cds^*} \geq F_{\bar{y}}$, then we can write:

$$\frac{Weight(cds)}{OPT_{cds^*}} \overset{(12)}{\leq} \frac{Weight(cds)}{F_{\bar{y}}} \overset{(18)}{\leq} 20 . \tag{19}$$

Let T be the gathering tree that produces Algorithm 3 and m is its MULE location. From Lemma 5 and

since the tree's BackBone is cds, we get:

$$Cost\,(T,m) \overset{(3)}{\leq} \sum_{v \in V_{BB}} (2 \cdot dist\,\{m,v\} + C) = \sum_{v \in cds} (2 \cdot dist\,\{m,v\} + C) = Weight\,(cds) \ . \tag{20}$$

Let us distinguish between the two MULE locations, m is the one chosen by the algorithm and m^* of the optimal solution. Let cds_m^* be the optimal solution of MWCDS for location m, where cds_m is the CDS constructed by the algorithm, and $cds_{m^*}^*$ is for location m^*. The algorithm selects the minimal-weight CDS, so we have the following inequality:

$$20 \cdot OPT_{cds_{m^*}^*} \overset{(19)}{\geq} Weight\,(cds_{m^*}) \geq Weight\,(cds_m) \ . \tag{21}$$

Let T^* be the optimal gathering tree and m^* is its MULE location. From Theorem 7, $OPT_{T^*} > (1+\varepsilon)^{-1} \cdot OPT_{cds_{m^*}^*}$. So we can put it all together into one equation and conclude that:

$$\frac{Cost\,(T,m)}{OPT_{T^*}} \overset{(9)}{<} \frac{Cost\,(T,m)}{(1+\varepsilon)^{-1} \cdot OPT_{cds_{m^*}^*}} \overset{(20)}{\leq} (1+\varepsilon) \cdot \frac{Weight\,(cds_m)}{OPT_{cds_{m^*}^*}} \overset{(21)}{\leq}$$

$$\overset{(21)}{\leq} (1+\varepsilon) \cdot \frac{Weight\,(cds_m)}{20^{-1} \cdot Weight\,(cds_m)} = 20 \cdot (1+\varepsilon) = 20 + \varepsilon \ ,$$

where the last equality is if we include the multiplication by 20 into the inaccuracy ε. So that

$$\varepsilon \triangleq 20 \left(\frac{2}{(C+2.6)} \cdot \left(\underset{v \in V_{BB}^*}{Average}\,[dist\,\{m^*,v\}] - 1.3 \right) \right)^{-1} . \qquad \qquad \qquad \square$$

Lemma 33. *Given a connected UDG with **a diameter of less than three**, Algorithm 3 returns a $(20 + \varepsilon) - approximate$ gathering tree to the $(1,1) - MULE - UDG$ problem.*

Proof. Since the graph is an UDG with a bounded area, then for each node $v \in V$ holds $dist\,\{m,v\} \leq 2$. Recall that the node weight function is $2 \cdot dist\,\{m,v\} + C$, so we have $Weight\,(cds) \leq (4+C) \cdot |cds|$. From Lemma 31, we know that $|cds| \leq 3 \cdot |ds|$. Therefore, we can write the following equation:

$$Weight\,(cds) \leq 3 \cdot (4+C) \cdot |ds| \ . \tag{22}$$

Note that $w\,(v) \geq C$. Let $mcds^*$ be the optimal solution for the MCDS problem. Thus, since cds^* is a CDS, it must be hold that $|cds^*| \geq |mcds^*|$. We obtain the following inequality:

$$OPT_{cds^*} = \sum_{v \in cds^*} w\,(v) \geq C \cdot |cds^*| \geq C \cdot |mcds^*| \ . \tag{23}$$

Let mis be a maximal independent set. Weili Wu et al. [13] showed that for UDG graph, $|mis| \leq 3.8 \cdot |mcds^*| + 1.2$. From Proposition 17, ds is a MIS, then

$$|ds| \leq 3.8 \cdot |mcds^*| + 1.2 \ . \tag{24}$$

If we sum it all up here, we get:

$$\frac{Weight\,(cds)}{OPT_{cds^*}} \overset{(22)}{\leq} \frac{3 \cdot (4+C) \cdot |ds|}{OPT_{cds^*}} \overset{(23)}{\leq} \frac{3 \cdot (4+C) \cdot |ds|}{C \cdot |mcds^*|} \overset{(24)}{\leq}$$

$$\overset{(24)}{\leq} \frac{3 \cdot (4+C) \cdot (3.8 \cdot |mcds^*| + 1.2)}{C \cdot |mcds^*|} =$$

$$= \frac{45.6 \cdot |mcds^*| + 11.4 \cdot C \cdot |mcds^*| + 14.4 + 3.6 \cdot C}{C \cdot |mcds^*|} \ .$$

Recall that $C > 14.5$, and $mcds^*$ is a DS, so it must contain at least one node, i.e. $|mcds^*| \geq 1$. We can continue the inequality as follows:

$$\frac{Weight\,(cds)}{OPT_{cds^*}} < \frac{45.6}{14.5} + 11.4 + \frac{14.4}{14.5 \cdot 1} + \frac{3.6}{1} < 19.14 < 20 \ . \tag{25}$$

As in Lemma 32, let us distinguish between two MULE locations, m and m^*. Remember that the algorithm selects the minimal-weight CDS, then we can state that:

$$20 \cdot OPT_{cds^*_{m^*}} \overset{(25)}{>} Weight\,(cds_{m^*}) \geq Weight\,(cds_m) \ . \tag{26}$$

Similarly to Lemma 32, we can put it all together and conclude that:

$$\frac{Cost\,(T, m)}{OPT_{T^*}} \overset{(9)}{<} \frac{Cost\,(T, m)}{(1+\varepsilon)^{-1} \cdot OPT_{cds^*_{m^*}}} \overset{(20)}{\leq} (1+\varepsilon) \cdot \frac{Weight\,(cds_m)}{OPT_{cds^*_{m^*}}} \overset{(26)}{<}$$

$$\overset{(26)}{<} (1+\varepsilon) \cdot \frac{Weight\,(cds_m)}{20^{-1} \cdot Weight\,(cds_m)} = 20 \cdot (1+\varepsilon) = 20 + \varepsilon \ ,$$

where the last equality is if we include the multiplication by 20 into the inaccuracy ε (as in Lemma 32). $\quad\square$

The total time complexity can be optimized to $O\left(n^3 \cdot \Delta\,(G)\right)$.

Proposition 34. *The number of active subsets maintained by Algorithm 1 is at most twice the number of nodes.*

Proof. Initially, the number of *active* subsets $(|\mathscr{S}'|)$ is as the number of nodes in the graph. We will show that for each node, at most one new subset is added to $\mathscr{S}'$. Without loss of generality, look at some node $u \in V$. Consider the first iteration in which the *if* statement in line 12 is met, for node u. Note, u is not yet selected by the algorithm. The new subset S' is added in line 14, where u is its only neighbor. Since the assignment $g\,(S') \leftarrow 1$ is performed, then only after the algorithm will assign $g\,(S') \leftarrow 0$, the statement will be able to met again. The only place where the assignment $g\,(S') \leftarrow 0$ can be performed is in line 9, when u is chosen by the algorithm. In the same step, node u is added to A, so the *if* statement in line 12 will not be checked again for u. Hence, no other new subset will be added for node u. $\quad\square$

Proposition 35. *At each step of Algorithm 2, a node can be contained in only one un-restricted active subset.*

Proof. Without loss of generality, let us consider node $u \in V$. We will prove by contradiction, if node u is contained in some *un-restricted* subset $S \in \mathscr{S}''$ then there is no other *un-restricted* subset $S' \in \mathscr{S}''$ that contains u. Let us assume in contradiction that u is contained in both S and S'. Note, u can not be added to the two subsets in the same step. We assume that u is already contained in subset S', and let us consider the step where u is added to subset S. According to propositions 22 and 23, since u is contained in S' then it cannot be the selected node. Therefore, u had to be added to S due to the union in line 18 of collection $\mathscr{S}^1$. S' is the only *un-restricted* subset containing u, so S' must be in the collection. By Proposition 23, the selected node v can not be in S'. Since $S' \in \mathscr{S}^1$, then there must be a neighbor of v is S'. Therefore, the *if* statement in line 16 is met and the assignment $g\,(S') \leftarrow 0$ is performed. But this is in contradiction to the assumption that S' is *un-restricted*. Hence, subset S must be the only *un-restricted* subset that contains u. $\quad\square$

Proposition 36. *At each step of Algorithm 2, a node can have at most five adjacent un-restricted subsets.*

Proof. We will prove by induction on the steps. Initially, the subsets of $\mathscr{S}''$ are based on the ds nodes and they are all *un-restricted*. Since ds is an IS then the statement holds for base case, step 0. The induction hypothesis is that the statement holds for step $k > 0$, and we will prove that the statement still holds for

step $k + 1$. We will show that for each node $u \in V$ the condition is preserved. Let v be the selected node in step $k + 1$, and S be the new subset. If u is not adjacent to any subset of $\mathscr{S}^1$ and is not a neighbor of v, then the new subset S has no influence on u. If u adjacent to at least one subset of $\mathscr{S}^1$, then after the union in line 18 the number of *un-restricted* subsets adjacent to u can not increase. Consider the case where u is not adjacent to any subset of $\mathscr{S}^1$ but is neighbor of v. Each subset adjacent to u contains a neighbor of u. Note, all of these neighbors are independent, and are also independent in v. Therefore, the number of subsets adjacent to u can not be greater than five, including the new *un-restricted* subset S that contains v. Hence, the condition holds for each step, and the statement is correct. $\qquad \square$

Algorithm 1 can be implemented in such a way that will produce a solution in $O\left(n^2 \cdot \Delta\left(G\right)\right)$ time. In order to do so, we maintain for each node $v \in V$ two additional variables. $sum_y\left(v\right)$ will be maintained equivalent to the value of $\sum_{S \in \mathscr{S}' \,:\, v \in N(S)} y\left(S\right)$, and $sum_g\left(v\right)$ for the value of $\sum_{S \in \mathscr{S}' \,:\, v \in N(S)} g\left(S\right)$. These two types of variables will be used to calculate the potential function $\epsilon\left(\cdot\right)$ and the *if* statement of line 12, in constant time. $sum_y\left(\cdot\right)$ variables will be initialized to zero, and $sum_g\left(\cdot\right)$ variables will be initialized to their node degree since all subsets are initially *un-restricted*. The $sum_y\left(\cdot\right)$ values should be updated after changing the $y\left(S\right)$ values in line 8. To do so, we will add an update after this line, for each node $u \in N\left(S\right)$, the $sum_y\left(u\right)$ value will be increased by $g\left(S\right) \cdot \epsilon$. The $sum_g\left(\cdot\right)$ values should be updated in two places. If the algorithm performs the assignment $g\left(S\right) \leftarrow 0$ in line 9, when previously $g\left(S\right)$ value was 1, then $sum_g\left(u\right)$ should be decrease by 1 for each $u \in N\left(S\right)$. After the assignment $g\left(S'\right) \leftarrow 1$ in line 14, $sum_g\left(u\right)$ should be increased by 1. Now, to calculate the algorithm runtime, we will focus within the general *repeat* loop, on the *for* loop in line 7. All other components can be computed in $O\left(n\right)$ time, if we refer to the complement subsets only symbolically and we consider unsorted sets. Each operation on $N\left(S\right)$ is performed in $O\left(\Delta\left(G\right)\right)$ time, since a subset can have as many neighbors as a node has. Note that according to Proposition 34, $\left|\mathscr{S}'\right| \leq 2n$. The *for* loop can perform at most $2n$ iterations, as the number of subsets in $\mathscr{S}'$. By Proposition 18, the *repeat* loop can perform at most n iterations. Therefore, the total runtime of Algorithm 1 is $O\left(n^2 \cdot \Delta\left(G\right)\right)$.

Algorithm 2 can also be implemented in such a way that will produce a solution in $O\left(n^2 \cdot \Delta\left(G\right)\right)$ time. Also here (as Algorithm 1), we maintain for each node $v \in V$ the two variables $sum_y\left(v\right)$ (for $\sum_{S \in \mathscr{S}'' \,:\, v \in N(S)} y\left(S\right)$) and $sum_g\left(v\right)$ (for $\sum_{S \in \mathscr{S}'' \,:\, v \in N(S)} g\left(S\right)$), and in addition the variable $\mathfrak{s}\left(v\right)$. This variable will hold an *un-restricted* subset that contains v, and from Proposition 35 we know that this subset is the only one. The first two types of variables will be used to calculate the potential function $\epsilon\left(\cdot\right)$ and the *if* statement of line 7, in constant time. While the third will be used in lines 5, 6 and 16 to find all subsets adjacent to a node, in $O\left(\Delta\left(G\right)\right)$ time. Here, variables $sum_g\left(v\right)$ will be initialized to $\left|N\left(\{v\}\right) \cap ds\right|$ for each node $v \in V$, $sum_y\left(v\right)$ to zeros, and variables $\mathfrak{s}\left(v\right)$ will hold subset $\{v\}$ for each $v \in ds$ and $\emptyset$ for each $v \in V \setminus ds$. Variables $\mathfrak{s}\left(\cdot\right)$ should be updated after adding the new subset in line 19. For each node u in the new *un-restricted* subset S we assign $\mathfrak{s}\left(u\right) \leftarrow S$. The $sum_y\left(\cdot\right)$ values should be updated after changing the $y\left(S\right)$ values by the *for* loop in line 14. Thus, after line 17 we will insert their update. Let $\mathscr{S}^1_u$ be the collection of all *un-restricted active* subsets adjacent to node u. For each node $u \in V$ we find $\mathscr{S}^1_u$ using $\mathfrak{s}\left(\cdot\right)$, and need to increase $sum_y\left(u\right)$ by $\left|\mathscr{S}^1_u\right| \cdot \epsilon$. The $sum_g\left(\cdot\right)$ values should be recalculated after changing the $g\left(S\right)$ values in lines 16 and 19. So, for each node $u \in V$ we find $\mathscr{S}^1_u$ using $\mathfrak{s}\left(\cdot\right)$, and assign $sum_g\left(u\right) \leftarrow \left|\mathscr{S}^1_u\right|$. Let us calculate the algorithm's runtime. The main component of the runtime is the *repeat* loop, which according to propositions 22 and 23, performs at most n iterations. According to Proposition 36, $\left|\mathscr{S}^2\right| \leq \left|\mathscr{S}^1\right| \leq 5$. Then, within the *repeat* loop, no component requires more than $O\left(n \cdot \Delta\left(G\right)\right)$ time to calculate (also the maintenance operations we added for $sum_y\left(\cdot\right)$ and $sum_g\left(\cdot\right)$). Therefore, the total runtime is $O\left(n^2 \cdot \Delta\left(G\right)\right)$.

The time complexity of Algorithm 3 is $O\left(n^3 \cdot \Delta\left(G\right)\right)$. Stems from the fact that the algorithm runs algorithms 1 and 2, n times.

6 Empirical Results

In this section we present simulation results of Algorithm 3. The simulations were performed on random UDG graphs, in a model of uniform random distribution of nodes across a given square area. Each graph is characterized by the size of the square area on which it is spread, and by the average density of its nodes. The simulation was performed only on connected graphs. The MULE transmission range (R_M) is determined to be 0.2. Since the goal is to show the α-approximation ratio obtained by the algorithm, the MULE location for each simulation is predetermined, in order to reduce runtime. The MULE location is intuitively determined to be the closest node to the center of the given area. There are two interesting aspects to explore. One aspect, is the performance of the algorithm for increasing nodes density, on a given area. For this aspect, we prepared three sequences of random graphs with the same area. The second aspect is the performance of the algorithm for increasing area. For this aspect, we prepared three sequences of random graphs with the same density of nodes.

The first results we present in Figure 4 (a) are simulations for increasing nodes density. We chose three sizes of square area of 4, 16 and 36 square units. For each area and density, we ran five simulations of different random graphs. We have seen that these areas are sufficient to represent the general trend. First of all, in the results we can see a convergence trend of all the simulations to a constant approximation value, as the density increases. This can be explained by the fact that as the density of the nodes grows, the selection of nodes for *cds* is increasingly based on their geographical location in the area. After all, we are looking for a set that will dominate the entire area. For a fixed area, the position will remain the same. Another thing that can be observed is the convergence of the distribution of the results, as the area increases. The convergence of the distribution can be explained as follows. The set of nodes that the algorithm finds, as well as the optimal solution, must dominate the entire area as the density increases. So when the area increases, also the weights of the nodes far from the MULE, are increased respectively. In the end, both sets (*cds* and the optimal solution) are required to dominate the same area, so the small shifts to the right and left, less affect the overall cost. What affects more, is the need to have one dominating node in the unit disk around each node in the graph, and this is also required from the optimal solution.

The second results we present in Figure 4 (b) are simulations for increasing area. We chose three different densities of 2.5, 10, and 40. We have seen that they represent the general trend. In addition, we illustrated the value of the approximation inaccuracy $(1 + \varepsilon)$ resulting from the reduction to the MWCDS problem. Since we can not calculate its exact value because it depends on the optimal solution, then we used the *cds* nodes that the algorithm finds. We denote this estimator by $\hat{\varepsilon}$. As we explained in the previous simulation results, as the area grows, then the general form of the optimal solution must be more similar to the solution found by the algorithm. Since the optimal gathering tree also needs to span all the nodes in the graph, then the trend should also be similar, as the area increases. Therefore, the $\hat{\varepsilon}$ value calculated by the average of the *cds* nodes distances should behave similarly to ε. In the results, we can see the convergence of the approximation value to a constant. The explanation for this is as before, since both the algorithm's solution and the optimal solution should dominate the entire area, and if a node is selected slightly to the right or left relative to the optimal solution node, then the effect becomes negligible relative to the weight of nodes far from the MULE. Another thing that can be observed is convergence as the density increases. The explanation for this is, of course, from the results of the previous simulation, in which we saw this convergence explicitly. One last thing to note is the downward trend of the estimator $(1 + \hat{\varepsilon})$, which gives us an insight for the value of $(1 + \varepsilon)$. For the range of area sizes we used, we can already see that the approximation inaccuracy has a very reasonable value.

7 Conclusions and Future Work

In this work, we studied the use of data MULEs in order to gather data and increase information survivability in wireless sensor networks. We considered the MULE problem for a general UDG with a single MULE and only one failed sensor, and referred to this problem as $(1,1) - MULE - UDG$. We showed that with a reasonable assumption it can be reduced to a MWCDS problem with an approximation of $(1 + \varepsilon)$. Then, we
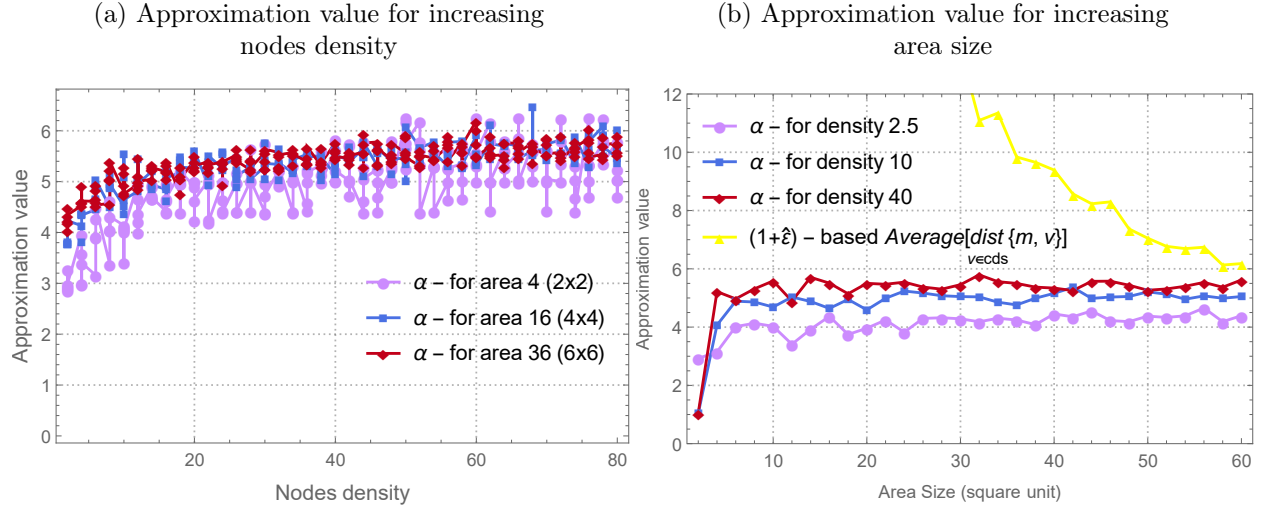
(a) Approximation value for increasing nodes density	(b) Approximation value for increasing area size

Figure 4: Simulation results for the approximation ratio α of Algorithm 3.

proposed a *primal-dual* algorithm that finds a $20-approximate$ solution to the reduced problem of MWCDS in polynomial time. Finally, we introduced the complete algorithm which produces a $(20 + \varepsilon)$-approximate solution for the $(1,1) - MULE - UDG$ problem in $O\left(n^3 \cdot \Delta\left(G\right)\right)$ time. Also, we have shown by simulation that in practice the algorithm achieves even better results. Future extensions of our work could investigate how to expand the algorithm for multiple MULEs in the network, or for different cost functions. In addition, it will be interesting to explore the implications of the algorithm technique on the research field of the MWCDS problem.

References

[1] Stav Ashur. Data gathering in faulty sensor networks using a mule. In *34th European Workshop on Computational Geometry (EuroCG '18)*, March 2018.

[2] Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1):165 – 177, December 1990.

[3] Jon Crowcroft, Liron Levin, and Michael Segal. Using data mules for sensor network data recovery. *Ad Hoc Networks*, 40:26 – 36, 2016.

[4] G. B. Dantzig, L. R. Ford, and D. R. Fulkerson. A primal-dual algorithm. May 1956.

[5] Thomas Erlebach and Matúš Mihalák. A (4+)-approximation for the minimum-weight dominating set problem in unit disk graphs. In *Approximation and Online Algorithms*, pages 135–146. Springer Berlin Heidelberg, 2010.

[6] M. Goemans and D. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[7] M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems, 1997.

[8] Liron Levin, Alon Efrat, and Michael Segal. Collecting data in ad-hoc networks with reduced uncertainty. *Ad Hoc Networks*, 17:71–81, jun 2014.

[9] Satish B. Rao and Warren D. Smith. Approximating geometrical graphs via "spanners" and "banyans". In *Proceedings of the thirtieth annual ACM symposium on Theory of computing - 98*. ACM Press, 1998.

[10] Rahul C. Shah, Sumit Roy, Sushant Jain, and Waylon Brunette. Data mules: modeling a three-tier architecture for sparse sensor networks. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, 2003.*, pages 30–41, May 2003.

[11] Arun A. Somasundara, Aditya Ramamoorthy, and Mani B. Srivastava. Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines. In *IEEE International Real-Time Systems Symposium*, number 25th. IEEE, December 2004.

[12] David P. Williamson. The primal-dual method for approximation algorithms. *Mathematical Programming*, 91(3):447–478, Feb 2002.

[13] Weili Wu, Hongwei Du, Xiaohua Jia, Yingshu Li, and Scott C.-H. Huang. Minimum connected dominating sets and maximal independent sets in unit disk graphs. *Theoretical Computer Science*, 352(1-3):1–7, mar 2006.

[14] Harel Yedidsion, Aritra Banik, Paz Carmi, Matthew J. Katz, and Michael Segal. Efficient data retrieval in faulty sensor networks using a mobile mule. In *2017 15th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, number 15th, pages 1–8. IEEE, may 2017.

[15] Feng Zou, Yuexuan Wang, Xiao-Hua Xu, Xianyue Li, Hongwei Du, Pengjun Wan, and Weili Wu. New approximations for minimum-weighted dominating sets and minimum-weighted connected dominating sets on unit disk graphs. *Theoretical Computer Science*, 412(3):198–208, jan 2011.