# Bipartivity Index based Link Selection Strategy to Determine Stable and Energy-Efficient Data Gathering Trees for Mobile Sensor Networks

Natarajan Meghanathan
Professor, Department of Computer Science, Jackson State University
Jackson, MS, USA
E-mail: natarajan.meghanathan@jsums.edu

**Keywords:** link stability, mobile sensor networks, bipartivity index, data gathering trees, simulations

**Received:** August 12, 2016

*Bipartivity Index (BPI) has been used in complex network analysis to quantify the extent of partitioning of the vertices of a network graph into two disjoint partitions; the edges between vertices within the same partition are called frustrated edges. The BPI values for a network graph ranges from 0 to 1 (the BPI of a network graph that is truly bipartite and has no frustrated edges is 1). Our hypothesis in this research is that the end nodes of a short distance link (the distance between the end nodes is significantly smaller than the transmission range per node) in a mobile sensor network (MSN) are more likely to share a significant fraction of their neighbors and such links are more likely to be stable. We introduce a notion called the egocentric network of an edge (adapted from egocentric network for a node) comprising of the end nodes of the edge and their neighbors (as vertices) and the edges incident on the end nodes (as edges). Our claim is that an edge whose egocentric network has a lower BPI score is more likely to be a stable short distance link, with a relatively larger fraction of shared neighborhood, and could be preferred for inclusion while determining stable data gathering trees for MSNs. Through extensive simulations, we show that the BPI-based DG trees are significantly more stable and energy-efficient compared to the DG trees determined using the predicted link expiration time (LET), currently the best known strategy.*

*Povzetek: Prispevek s pomočjo BPI indeksa ugotavlja stabilna in energijsko učinkovita drevesa za mobilne senzorske mreže.*

## 1 Introduction

Mobile Sensor Networks (MSNs) are an emerging category of wireless sensor networks in which the sensor nodes are considered to move independent of each other. MSNs could be used for applications in which an entire region (that is being monitored) could be effectively covered by letting the sensor nodes to move rather than be static. For example [9], the pollutant concentration in an area (like the downtown of a city) could be effectively measured by fixing the sensor nodes in mobile vehicles (like cars) that move through the area. For most of the applications of wireless sensor networks (including those of the MSNs), the data recorded by the sensor nodes is forwarded to a control center (called the sink) through one of several network-wide communication topologies (like chains [11], clusters [7], trees [18], connected dominating sets [16], etc). Among these communication topologies, the data gathering trees (DG trees) have been observed to be energy-efficient [18] as they comprise of the minimum number of links needed to span all the sensor nodes and there are no redundant transmissions. In the case of DG trees, the leaf nodes merely sense the data and transmit them to an upstream intermediate node that would in turn aggregate its own data with data received from all of its child nodes and forward the aggregated data to an upstream node that is on the path to the root node of the DG tree. For the rest of the paper, the terms 'node' and 'vertex', 'link' and 'edge', 'network' and 'graph', 'data gathering' and 'data aggregation', 'construction' and 'configuration' mean the same. These terms are used interchangeably unless stated.

MSNs inherit all the constraints of their static counterpart (like energy and memory-constrained sensor nodes as well as limited network bandwidth); mobility of the nodes is an additional constraint that needs to be handled. Due to node mobility, the network topology changes dynamically with time and any communication topology (like DG trees) that is setup among the sensor nodes needs to be frequently reconfigured. Significant amount of energy might be lost if network-wide broadcasts are frequently initiated for reconfiguring the communication topology in use. This motivates the need to determine stable communication topologies that could exist for a longer time.

In [19], the authors took the first step towards using DG trees for MSNs and proposed a distributed algorithm for determining stable DG trees in MSNs using the concept of predicted link expiration time (LET) [31] that has been earlier successfully used for mobile ad hoc

networks [20, 22]. In [24], the authors proposed a generic algorithm to determine maximum bottleneck link weight (MaxBLW)-based DG trees for static sensor networks: the bottleneck link weight for a path from a node to the root node of the DG tree is the minimum of the weights of the constituent links on the path and the MaxBLW-DG algorithm determines a DG tree in which the path from any node to the root node of the tree is the path with maximum value for the bottleneck link weight. In this paper, we explain a distributed version of the MaxBLW-DG algorithm to determine ALGC-based DG trees wherein the link weight is the link stability score (LSS) computed based on this strategy. For performance comparison purposes, we use the distributed version of the MaxBLW-DG tree algorithm to also determine the LET-based DG trees [19] wherein the weight of a link is its predicted LET.

The LET-based strategy is the only available link selection strategy that has been successfully demonstrated so far [19] for determining stable DG trees in MSNs. However, the LET formulation [19, 31] does not consider the distance between the constituent end nodes of a link and is prone to choosing links that could incur a larger transmission energy and ultimately contributing towards larger energy consumption per round. We opine that links whose constituent end nodes are closer to each other (i.e., the distance between the end nodes of the link is appreciably lower than the transmission range per node) are more likely to be stable (and vice-versa) as it would take a while for such end nodes to move out of the transmission range of each other. We refer to such links as "short distance" links. Also, as the energy lost per transmission is directly proportional to the square of the distance [28] over which the transmission is made, we claim that the DG trees comprising of short distance links are more likely to be both stable (and vice-versa) as well as incur lower energy consumption per round. Moreover, the LET approach [31] requires a sensor node to be aware of its own location and mobility as well as that of its neighbors. This would require the sensor nodes to be equipped with energy-draining hardware/software systems (like GPS [8]) that would make them location and mobility aware. All of the above observations form the motivation for the research conducted in this paper.

The high-level contribution of this paper is that we show the use of a spectral graph-theoretic metric called Bipartivity Index (BPI) [6] to quantify the extent of shared neighborhood between the end vertices of an edge and thereby model the link stability score (LSS) for the edge. The BPI has been widely used in complex network analysis [6] to quantify the extent of partitioning of a network graph into two disjoint partitions of vertices; the edges between vertices within the same partition are referred to as frustrated edges. BPI values range from 0 to 1 [6]. A network graph is said to be truly bipartite (such a partitioning also has no frustrated edges) if its BPI is 1 [6]. We propose to use a notion called the "egocentric network of an edge" (adapted from the notion of egocentric network of a node [13]) to quantitatively evaluate the extent of shared neighborhood between the end vertices of a link. The egocentric network of an edge $u$-$v$ (denoted $EG_{u\text{-}v}$) comprises as vertices - the end nodes of the edge and their neighbors, and edges - the links incident on the end nodes of the edge. We claim that an edge $u$-$v$ is more likely to be a stable short distance link with a larger fraction of shared neighborhood if the egocentric network $EG_{u\text{-}v}$ of the edge has a lower BPI. Accordingly, we model for an edge $u$-$v$: the LSS($u$-$v$) as 1 - BPI($EG_{u\text{-}v}$).

We provide a high-level justification for the above modeling as follows (more details are presented in Section 4). If the end nodes of an edge $u$-$v$ do not have any shared neighbors, then the egocentric network of the edge $u$-$v$ would comprise of node $u$ and the neighbors of node $v$ in one of the two partitions, and node $v$ and the neighbors of node $u$ in the other partition; all the edges would connect the vertices in one partition to the other partition and there would be no frustrated edges within either partition (a frustrated edge is an edge involving vertices that are in the same partition [6]). Such an egocentric network is truly bipartite and will have a BPI of 1. Whereas, if the end nodes of an edge $u$-$v$ have one or more shared neighbors, the egocentric network of the edge (when analyzed for bipartivity) would comprise of one or more frustrated edges contributing to a BPI less than 1. We anticipate the BPI for the egocentric network of an edge $u$-$v$ to reduce with increase in the number of shared neighbors for the end nodes $u$ and $v$.

The rest of the paper is organized as follows: Section 2 outlines the maximum bottleneck link weight-based algorithm for determining data gathering trees in sensor networks. Section 3 reviews related work, including the strategy of using the predicted link expiration time (LET) to determine stable data gathering trees for MSNs. Section 4 introduces the notions of short distance links, egocentric network for an edge and bipartivity index (BPI) as well as illustrates their use to quantify the extent of shared neighborhood and stability of links. Section 5 presents results of exhaustive simulations conducted to showcase the effectiveness of the BPI-based strategy to determine data gathering trees that are both stable as well as energy-efficient compared to the LET-based DG trees. Section 6 concludes the paper.

# 2 Distributed algorithm to construct a maximum bottleneck link weight-based data gathering tree

In this section, we describe a distributed version of the algorithm to construct maximum bottleneck link weight-based data gathering (MaxBLW-DG) trees for mobile sensor networks. A centralized version of the MaxBLW-DG algorithm has been earlier proposed in [24] and a distributed implementation of the algorithm to determine LET-based stable data gathering trees has been discussed in [19]. The distributed version of the MaxBLW-DG algorithm discussed here could be applied for any measure of link weight. For this section, we assume the link weights are randomly generated in the range [0...1].

In sections 4 and 5, the weight of a link depends on the link selection strategy (BPI or LET) employed.

## 2.1 Assumptions and definitions

We assume the sensor nodes to operate in a fixed transmission range, *R*. We assume the underlying network is modeled as a unit-disk graph wherein there exists a link between any two nodes if the Euclidean distance between them is within the transmission range, *R*. We assume the network to be homogeneous (i.e., all the nodes have an equal transmission range). We define the *fraction of link distance* (*fld*) as the ratio of the Euclidean distance between the end nodes of the link and the transmission range per node. In the case of heterogeneous networks (each node operating with a different transmission range), the fraction of link distance could be measured as the ratio of the Euclidean distance between the end nodes of the link and the maximum of the transmission ranges of the two end nodes. The data gathering algorithms (discussed in Sections 2 and 3) and the BPI strategy discussed in Section 4 could be used for both homogeneous and heterogeneous networks. For a directed edge $u \rightarrow v$, we refer to node $u$ as the upstream node and node $v$ as the downstream node. In the context of link weights, we assume the links/edges are undirected (bidirectional): i.e., the weight of a directed edge $u \rightarrow v$ is the same as the weight of the directed edge $v \rightarrow u$.

We define a round of data gathering to comprise of steps in which the sensor nodes individually sense the data within their sensing range (typically the sensing range of a sensor node is at most half its transmission range [36]), aggregate and forward only a representative version of the data (like the average temperature in a region) to the sink through a network-wide communication topology (like a data gathering tree) spanning all the sensor nodes. The size of the aggregated data is assumed to be the same as the size of the data collected at the individual sensor nodes. The root node of a DG tree is called the LEADER node and is chosen by the sink at the time of tree construction.

We assume the sensor nodes to be both TDMA (Time Division Multiple Access) and CDMA (Code Division Multiple Access)-enabled [35]. An upstream node communicates with its own immediate downstream child nodes using a TDMA schedule (one time slot per downstream node); such communication between every upstream node with their own downstream nodes can occur in parallel (using unique CDMA codes). The above assumptions and definitions hold good for both the BPI and LET-based MaxBLW-DG trees studied in this paper.

When used for constructing the LET-based DG trees, we assume a sensor node to be aware of its current location, velocity and direction of movement at any time instant and mentions the same in a location update vector (LUV) [19] included in the control messages broadcast as part of tree discovery. Such an assumption is not required for the MaxBLW-DG algorithm that makes use of BPI (discussed in Section 4) as the BPI scores could be computed without a priori knowledge about the location and mobility of the nodes. Each node maintains a *Link Weight Table* comprising of the estimates of the bottleneck link weights to the neighbor nodes that sent it the TREE-CONSTRUCT message (see Section 2.3 for more details about the message).

## 2.2 Initialization of state information at the sensor nodes

Each sensor node locally maintains state information about the data gathering tree that is currently being used or newly configured. The state information comprises of the following fields (with their initial values indicated in parenthesis): *estimated bottleneck link weight* (-∞), *upstream node id* (NULL), *tree level* (0), *LEADER node id* and *sequence number* (the latest sequence number in the TREE-INITIATE message broadcast by the sink). The *estimated bottleneck link weight* is the value for the currently known maximum weight for a link on the path to the LEADER node of the DG tree. The *upstream node id* corresponds to the neighbor node that lies on the currently estimated maximum bottleneck link weight path to the LEADER node. The *tree level* corresponds to the number of hops on the maximum bottleneck link weight path to the LEADER node. The *sequence number* corresponds to the latest sequence number for a TREE-CONSTRUCT message received by the node.

## 2.3 Initiation of the Tree-construct message

Whenever the sink fails to receive the aggregated data from the LEADER node of the DG tree used in the previous round of data aggregation, the sink queries all the sensor nodes to send it their estimates of the weight of the links from their neighbor nodes. The sink calculates the estimated weight of a node as the sum of the estimated weights of the directed edges originating from the node (as reported by its neighbors); the sink selects the node with the largest estimated weight to be the LEADER node (root node) of the new DG tree that is to be setup and sends a TREE-INITIATE message (including a sequence number) to the chosen root node to begin the construction of the new DG tree. The sequence number for the tree construction process is a monotonically increasing value maintained at the sink and the sink sends the latest value of the sequence number to the LEADER node to facilitate the sensor nodes to uniquely identify the control messages that are exchanged with regards to the new DG tree being constructed.

The LEADER node broadcasts a TREE-CONSTRUCT message to its neighbors; the message has a 5-element tuple: <sequence number, LEADER node id, upstream node id, sender's estimated bottleneck link weight, tree level>. The sequence number is the one that is sent by the sink to the LEADER node. For the TREE-CONSTRUCT message broadcast by the LEADER node, the values for the upstream node id, sender's estimated bottleneck link weight and tree level are respectively the LEADER node id, +∞ and 0. For the TREE-CONSTRCT message broadcast by the other nodes: the upstream node

id is the id of the node that the sender of the message considers to be the best node that would connect it to the LEADER node through a path estimated to be the one with the maximum bottleneck link weight (the value of which is also indicated in the message). The tree level field indicates the number of hops on the estimated maximum bottleneck link weight path from the sender node to the LEADER node of the DG tree.

### 2.4 Propagation of the Tree-construct message

When a node receives the TREE-CONSTRCT message (from a neighbor node) with a higher sequence number, it assumes that the DG tree that had been used until then no longer exists and resets its state information to the values listed in Section 2.2. The receiving node (say, node *v*) decides to further process the TREE-CONSTRUCT message from a neighbor node (say, node *u*) if all the following conditions are met: (i) The *upstream node id* in the message is different from the id of the receiving node itself. (ii) The *tree level* value in the message is less than or equal to the *tree level* value maintained as part of the state information at the receiving node. (3) The value for the *sender's estimated bottleneck link weight* in the message is larger than the value for the receiver's estimated bottleneck link weight. (4) The weight of the directed edge from the sender node to the receiver node is larger than the latter's estimated bottleneck link weight for the path to the LEADER node. If all the above four conditions are met, the receiver node (node *v*) makes the following updates to its state information: (i) The receiver node updates its *estimated bottleneck link weight* for the path to the LEADER node to the minimum of the sender's (node *u*'s) estimated bottleneck link weight value in the TREE-CONSTRUCT message and the weight of the directed edge $u \rightarrow v$. (ii) The receiver node updates its *upstream node id* to that of the sender's node id. (iii) The value for the *tree level* is set to one more than the value for the tree level in the TREE-CONSTRUCT message. After making the above updates, the receiver node also rebroadcasts the TREE-CONSTRUCT message in its neighborhood by changing the values for the *upstream node id*, *sender's estimated bottleneck link weight* and *tree level* fields in the message to the most recently updated values for these fields in its state information.

Overall, a node receiving the TREE-CONSTRUCT message decides to further rebroadcast the message only if it can increase (through the sender node that sent it the message) its estimate for the bottleneck link weight path to the LEADER node (thus minimizing unnecessary retransmissions). Each node (other than the LEADER node) will be able to do so at least once because its initial value for the estimated bottleneck link weight is $-\infty$ and all edge weights are positive as well as the value for the sender's estimated bottleneck link weight in the TREE-CONSTRUCT message broadcast by the LEADER node is $+\infty$. At the end of the tree construction process, each node (other than the LEADER node) would have joined the DG tree through an upstream node that is on the maximum bottleneck link weight path to the LEADER node.

### 2.5 Propagation of the Tree-link-failure message

Whenever an upstream node fails to receive an aggregated data packet from one of its downstream child nodes, the upstream node decides that the link to the child node has broken and initiates a TREE-LINK-FAILURE message (included with a sequence number corresponding to the value sent by the LEADER node in the TREE-CONSTRUCT message) with the number of hops the message can get propagated equal to the tree level value for the initiating upstream node. The TREE-LINK-FAILURE message is essentially reverse broadcast higher up the currently used DG tree so that the LEADER node can receive the failure message and initiate the construction of a new DG tree. Nodes that lie downstream of the failed link get to learn about the tree failure when a TREE-CONSTRUCT message with a higher sequence number (larger than the current value for the sequence number known) is received.

## 3 Related work

In this section, we first discuss related work data gathering in mobile sensor networks and then focus our discussion specifically on related work on determining stable data gathering trees in mobile sensor networks.

### 3.1 Related work on data gathering in mobile sensor networks

To the best of our knowledge, other than the work presented in Section 3.2 and the related works discussed below, the existing works (e.g., [10, 14, 32, 33]) in the literature on mobile sensor networks take the following hybrid approach: The regular data sensing nodes are considered static and there exists one or more mobile data collecting nodes that move around the static sensor nodes; a data gathering topology involving the data collecting nodes is constructed and maintained, if needed. Since all the sensor nodes are not considered mobile (the type of mobile sensor networks considered in our research) and the data gathering topology constructed is not network-wide (i.e., spanning all the sensor nodes), we do not delve further on related works based on the above approach.

Among the very few network-wide spanning topology-based data gathering algorithms available in the literature for mobile sensor networks, most of the work focused on extending the classical LEACH (Low Energy Adaptive Clustering Hierarchy) [7] algorithm for static sensor networks to adapt to mobile environments. Variants of LEACH that have been proposed for MSNs focus on choosing the cluster heads by taking into account the residual energy available at the sensor nodes [2], mobility of the sensor nodes [29], stability of the links incident on a node [5] or proximity of the sensor nodes to certain landmarks [12]. Another work [30] related to cluster head selection proposed to set up a

panel of cluster heads (some of which serve as backup) to facilitate cluster reconfiguration due to node mobility.

In [34], the authors proposed a cluster independent data collection tree (CIDT) protocol for mobile sensor networks that first partitions the entire network into clusters with a cluster head plus member nodes for each cluster and then chooses certain sensor nodes as data collection nodes (DCNs) that have better connection with the cluster heads. A data gathering tree of the DCNs is constructed and reconfigured over time when broken due to node mobility. The DCNs are selected in such a way that the links to the cluster heads and the links to the adjacent DCNs in the data gathering tree are stable. We opine that the CIDT protocol would incur a lot of control overhead (with respect to bandwidth and energy consumption) as two topologies (a cluster topology comprising of cluster heads plus their links to the member nodes and a tree topology of DCNs) have to be maintained in the network at any time. Though the two topologies have been formulated to be independent of each other, (due to node mobility) the identification of cluster heads and the DCNs has to be often initiated to maintain connectivity of the cluster heads to one or more near by DCNs.

In [15], the authors propose a directed acyclic graph (DAG)-based topology for determining data gathering trees in mobile sensor networks. Whenever a data gathering tree is required, the sink constructs a DAG of the underlying network and runs a maximum bottleneck node weight-based data gathering (MaxBNW-DG) algorithm on the DAG. In this pursuit, the sink initiates data collection from all the nodes in the network on one or more multi-hop paths; the paths traversed by the data in cycle-free manner constitute a DAG of the network. The weight of a sensor node is determined based on the theory of thermal fields applied on the utility of the data sensed by the node as well as that of its neighbors. The sink then initiates a distributed version of the MaxBNW-DG algorithm on the DAG such that each sensor node is located on a maximum bottleneck node weight path to the sink node. The bottleneck node weight for a path in [15] is calculated as the minimum of the weights of the intermediate node on the path; ties are broken in favor of paths of lower hop count. Due to node mobility, there may not be paths from one or more nodes to the sink node on the DAG. Similar to [34], we opine that a significant control overhead (in a mobile sensor network) would be encountered to first construct a DAG and then run a distributed version of the MaxBNW-DG algorithm on the DAG.

## 3.2 Related work on stable data gathering trees in mobile sensor networks

In [21], the authors had proposed a benchmarking algorithm to determine a sequence of stable data gathering trees that would exist for the longest time such that the number of tree transitions is the bare minimum. When a DG tree is required at a time instant *t*, the idea is to determine an intersection of the network graphs existing at time instants *t*, *t*+1, *t*+2, ... *t*+*k* such that the intersection graph is connected from time instants *t* ... *t*+*k* and not connected from time instants *t* ... *t*+*k*+1. That is, the inclusion of the graph at time instant *t*+*k*+1 to the intersection graph of time instants *t* ... *t*+*k* would disconnect the intersection graph from time instants *t* ... *t*+*k*+1. However, the algorithm is centralized in nature and would require the topology changes to be known a priori from the beginning to the end of the simulation session. On the other hand, the focus of research in this paper is to employ a distributed algorithm for determining stable data gathering trees using the BPI approach from complex network analysis - this approach does not require any a priori knowledge about the network topology changes as well as about the location and mobility of the nodes; we would just need the one-hop neighborhood information at every node.

In [19], the authors proposed distributed algorithms to determine the predicted link expiration time (LET)-based data gathering trees for longer tree lifetime and the minimum distance spanning tree (MST)-based data gathering trees for longer node lifetime (time of first node failure due to exhaustion of energy) and longer network lifetime (time at which the network gets disconnected due to the failure of one or more nodes). The predicted link expiration time (LET) of a link $i - j$ between two nodes $i$ and $j$, currently at ($X_i$, $Y_i$) and ($X_j$, $Y_j$), and moving with velocities $v_i$ and $v_j$ in directions $\theta_i$ and $\theta_j$ (with respect to the positive X-axis) is computed using the formula proposed in [31]:

$$LET(i,j) = \frac{-(ab+cd) + \sqrt{(a^2+c^2)R^2 - (ad-bc)^2}}{a^2+c^2} \quad (1)$$

where $a = v_i*\cos\theta_i - v_j*\cos\theta_j$; $b = X_i - X_j$;
$c = v_i*\sin\theta_i - v_j*\sin\theta_j$; $d = Y_i - Y_j$

The MST-based DG trees aim to minimize the largest Euclidean distance between the end nodes of a link in the DG tree, but are not as stable as the LET-DG trees [19]. Due to repeated tree reconfigurations, the gain obtained in the node lifetime (85-150% more than that of the LET-DG trees) does not equally get transferred to the gain obtained in network lifetime (only 15-130% more than that of the LET-DG trees) [19]. The LET-DG trees fit within the criteria of finding maximum bottleneck link weight-based DG trees (i.e., the objective is to maximize the minimum LET for a link on the path from any node to the LEADER node); whereas, the MST-DG trees fit within the criteria of finding minimum bottleneck link weight-based DG trees (i.e., the objective is to minimize the maximum value for the distance between the end nodes of the link on the path from any node to the LEADER node). In this paper, we model the short distance links as links with larger BPI/link stability score (measure of link weight; for further details, see Section 4) and run the maximum bottleneck link weight-based algorithm to maximize the minimum link weight on the path from any node to the LEADER node of the DG tree; we show that by doing so, we can simultaneously incur a larger tree lifetime as well as a lower energy

consumption per round (see Section 5 for the simulation results).

In [24], the authors proposed a generic algorithm to determine maximum bottleneck node weight (MaxBNW)-based data gathering trees wherein the root node is the node with the largest weight. In [24], the weight of a node has been modeled as the sum of the weights of the links incident on it. The bottleneck node weight for a path from a node to the root node of the DG tree is the minimum of the weights of the nodes on the path. The MaxBNW-DG algorithm aims to determine a DG tree in which the path from any node to the root node is the path with the maximum bottleneck node weight. In [24], it has been observed that the MaxBNW-DG trees have different characteristics compared to the MaxBLW-DG trees. The focus of research in this paper is to determine MaxBLW-DG trees by modeling the link weight as a measure of the stability of the link using the algebraic connectivity approach from complex network analysis that does not need the location and mobility information of the nodes.

# 4 Bipartivity index (BPI)-based link selection strategy

In this section, we describe the Bipartivity Index (BPI) [6]-based link selection strategy adapted from complex network analysis to quantify the stability of links (i.e., the link weights) in a mobile sensor network. We compute the link stability score (LSS) for an edge by analyzing the bipartivity of the "egocentric network for the edge" that is adapted from the notion of egocentric network of a node [13]. The *egocentric network of a node* [13] in a graph is a sub graph comprising of: *vertices* - the nodes and its neighbors and *edges* - the links involving the node and/or its neighbors. We define the *egocentric network of an edge* in a graph to be a sub graph comprising of: *vertices* - the end nodes of the edge and their neighbors and *edges* - the links incident on the end nodes of the edge.

Our hypothesis for this research is based on the observation that links (we refer to as *short distance* links) whose end nodes are close enough to each other (vis-a-vis the transmission range per node) are more likely to be stable (and vice-versa) compared to links for which the distance between the end nodes is closer to the transmission range per node. We define the *fraction of link distance* (*fld*) for an edge as the ratio of the Euclidean distance between the end nodes of the edge and the transmission range per node. For a short distance link, *fld* is expected to be appreciably less than 1. Our hypothesis is that the end nodes of a short distance link are more likely to share a significant fraction of their neighbors (and vice-versa) and we could compute the BPI for the egocentric network of the link to quantify the extent of this shared neighborhood that can be in turn used as the link stability score (LSS). Note that the egocentric network of an edge could be independently (and identically) constructed by each of the two end nodes of the edge based on the one-hop neighborhood information received from the other node (as part of periodic beacon exchange).

A graph is said to be truly bipartite [4, 6] if we could partition the vertices of the graph into two disjoint sets such that all the edges in the graph are those that connect the vertices in one partition to vertices in the other partition and that there are no edges (called frustrated edges [6]) between vertices within the same partition. However, all network graphs cannot be expected to be truly bipartite. Hence, Estrada and Rodriguez-Velazquez [6] proposed the notion of bipartivity index (BPI) to measure the extent of bipartivity in a graph. The bipartivity index of a graph ranges from 0 to 1. If a graph is truly bipartite, then the bipartivity index is 1 and there are no frustrated edges between vertices within the same partition [6]. If a graph is not truly bipartite, then the bipartivity index will be less than 1. Estrada and Rodriguez-Velazquez [6] proposed a mechanism that will allow us to identify a partitioning of the vertices into two disjoint partitions as well as identify the frustrated edges (if the graph is not truly bipartite) involving vertices within the same partition. The mechanism proposed by Estrada Rodriguez-Velazquez [6] is to determine the eigenvalues of the adjacency matrix of the graph (to determine the bipartivity index, as shown in formulation 2) and use the signs (positive or negative) of the entries in the eigenvector corresponding to the smallest eigenvalue of the adjacency matrix to determine the partitioning of the vertices. If $\lambda_1$, $\lambda_2$, $\lambda_3$, ..., $\lambda_n$ are the eigenvalues of the adjacency matrix of a graph $G$ of $n$ vertices, then the bipartivity index (BPI) of $G$ is given by the formulation below [6]:

$$BPI(G) = \frac{\sum_{j=1}^{n} \cosh(\lambda_j)}{\sum_{j=1}^{n} \cosh(\lambda_j) + \sum_{j=1}^{n} \sinh(\lambda_j)} \quad (2)$$

To measure the extent of shared neighborhood of the end vertices of an edge in a graph, we propose to compute the bipartivity index on the egocentric network of the edge and use the complement of the bipartivity index (1 - BPI) as the link stability score (LSS) for the edge. That is: LSS($u$-$v$) = 1 - BPI($EG_{u\text{-}v}$), where $EG_{u\text{-}v}$ is the egocentric network graph of the edge $u$-$v$. We justify the above proposal as follows (also illustrated in Figures 1-3: for an edge $u$-$v$ where $u$1-$u$4 are four neighbors, other than vertex $v$, for a vertex $u$; and $v$1-$v$4 are four neighbors, other than vertex $u$, for a vertex $v$):

If the end vertices of an edge $u$-$v$ do not have any shared neighbors (i.e., $u$1-$u$4 and $v$1-$v$4 are all distinct vertices: as shown in Figure 1), then we could partition the vertices in the egocentric network of the edge to two disjoint partitions such that vertex $u$ and the neighbors of vertex $v$ are in one partition (referred to as *partition-u*) and vertex $v$ and the neighbors of vertex $u$ are in the other partition (referred to as *partition-v*). The egocentric network of the edge $u$-$v$ with no common neighbors for the end vertices would be a truly bipartite graph (as in

Figure 1) as the only edges in the graph would be edges connecting vertices in *partition-u* to vertices in *partition-v*. The bipartivity index of such an egocentric network graph would be 1.0 and as per our hypothesis, the link stability score for the edge would be 0.0.

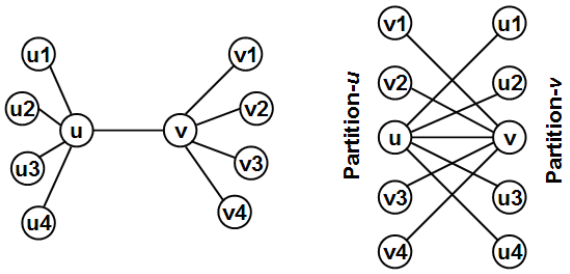
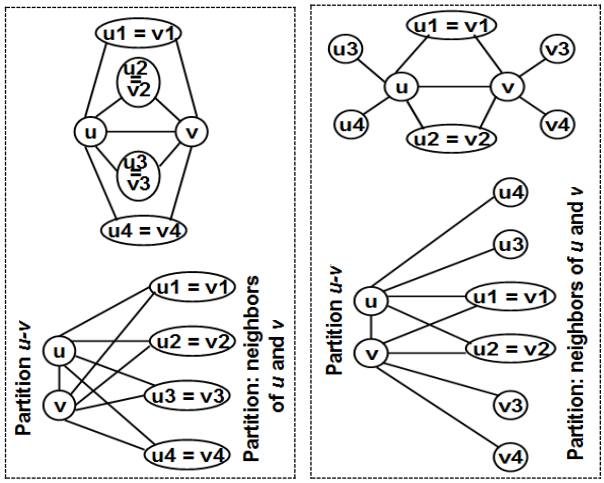
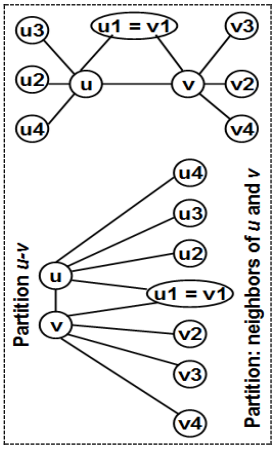



Figure 1: Example for a Truly Bipartite Egocentric Network of an Edge *u-v*.



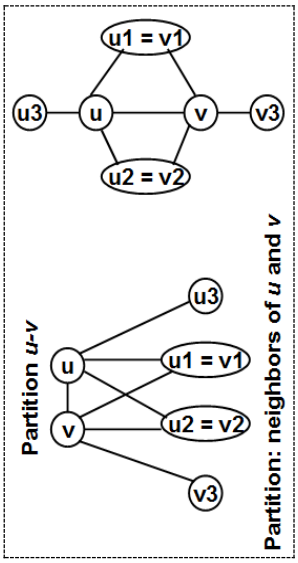

2-a: BPI($EG_{u\text{-}v}$) = 0.75
LSS($u$-$v$) = 0.25

2-b: BPI($EG_{u\text{-}v}$) = 0.85
LSS($u$-$v$) = 0.15

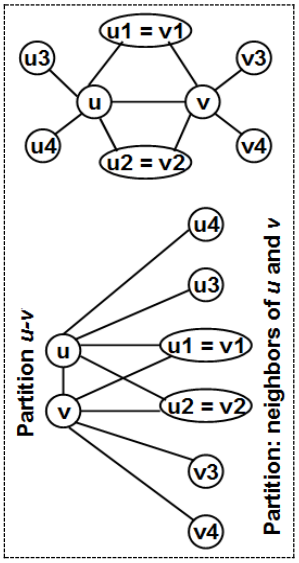



2-c: BPI($EG_{u\text{-}v}$) = 0.92
LSS($u$-$v$) = 0.08

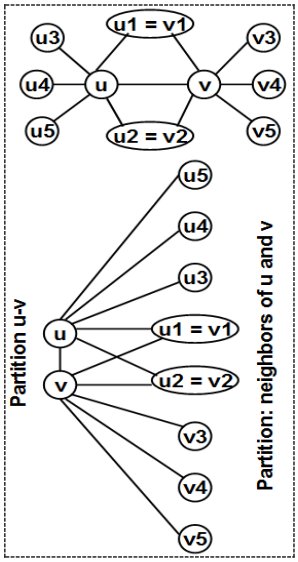
Figure 2: Bipartivity Index of the Egocentric Network Graph of an Edge and its Link Stability Score (Varying the Number of Shared Neighbors for a Fixed Number of Edges in the Egocentric Network).




3-a: BPI($EG_{u\text{-}v}$) = 0.83
LSS($u$-$v$) = 0.17




2-b: BPI($EG_{u\text{-}v}$) = 0.85
LSS($u$-$v$) = 0.15




3-c: BPI($EG_{u\text{-}v}$) = 0.87
LSS($u$-$v$) = 0.13

Figure 3: Bipartivity Index of the Egocentric Network Graph of an Edge and its Link Stability Score (Varying the Number of Edges in the Egocentric Network for a Fixed Number of Shared Neighbors).

On the other hand, if the end vertices of an edge *u-v* share one or more of their neighbors: then, the eigenvector-based decomposition for bipartivity [6] applied on the egocentric network of the edge would group the two end vertices *u* and *v* together in one partition (referred to as *partition u-v*) and the neighbors of *u* and *v* together in the other partition (referred to as *partition: neighbors of u and v*). The BPI of such egocentric network graphs would be less than 1 (due to the presence of the frustrated edge *u-v* in the same partition) and the actual magnitude of the BPI would depend on the actual number of neighbors for the two end vertices (i.e., on the number of vertices in the other

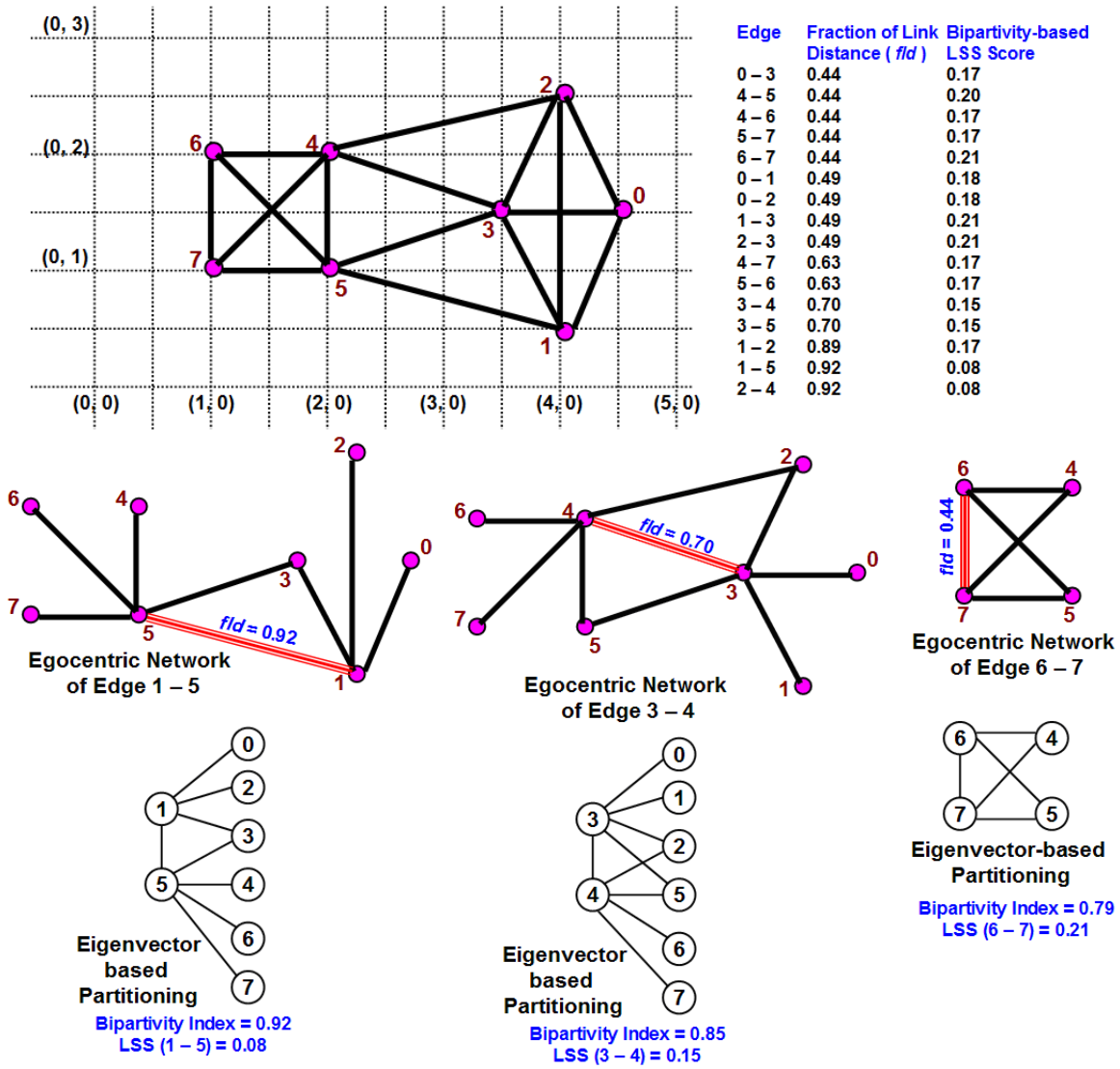



Figure 4: Illustration of the Eigenvector-based Partitioning of the Egocentric Networks of the Edges and the Bipartivity Index and Link Stability Scores of the Edges in an Example Graph.

*partition: neighbors of u and v*) as well as on the number of shared neighbors (i.e., on the number of edges connecting the vertices in *partition u-v* to the vertices in the *partition: neighbors of u and v*). For a given egocentric network graph $EG_{u\text{-}v}$ with a certain number of edges and is not truly bipartite (as shown in Figures 2-a, 2-b and 2-c): the larger the number of shared neighbors (i.e., fewer the number of vertices in the *partition: neighbors of u and v*), the lower the BPI (and larger will be the LSS score for the edge *u-v*). Likewise, for a given egocentric network graph $EG_{u\text{-}v}$ with a certain number of shared neighbors and is not truly bipartite (as shown in Figures 3-a, 3-b and 3-c): the larger the number of vertices in the *partition: neighbors of u and v* (i.e., the larger the number of edges connecting the vertices in *partition u-v* to the vertices in the *partition: neighbors of u and v*), the larger the BPI (and lower will be the LSS score for the edge *u-v*).

In Figure 4, we illustrate the computation of the bipartivity index of the egocentric networks of the edges (with coordinates of the vertices as indicated in a grid) in an example graph. The egocentric networks for none of the edges have been observed to be truly bipartite. We illustrate the eigenvector-based partitioning of the egocentric networks of three edges: 6-7, 3-4 and 1-5 that have different values for the fraction of link distance (*fld*). We observe the bipartivity-based LSS values (1 - bipartivity index) for these three edges to increase with decrease in the fraction of link distance (*fld*) values. Overall, we see the expected trend between *fld* and the bipartivity-based LSS values for the edges: the LSS values are more likely to be higher for edges with lower *fld* values and vice-versa.

# 5 Simulations

In this section, we first present the simulation environment and the notion of normalized comprehensive relative performance (NCRP) score to identify the link selection strategy that effectively balances the tradeoffs with respect to the performance metrics, and then discuss in detail the results of the

simulations obtained by running the distributed version of the MaxBLW-DG algorithm incorporated with BPI as well as the LET-based link selection strategies. The simulations were conducted in a discrete-event simulator implemented in Java for mobile sensor networks. The simulator was earlier successfully used for other related studies (e.g., [19][21][23]) for mobile sensor networks. The medium access control (MAC) layer is assumed to be ideal to extract the best possible performance from the data gathering algorithm and the link selection strategies.

## 5.1 Simulation environment

In this sub section, we present the simulation parameters (network density, maximum node velocity, data size and the number of rounds as well as the frequency of LSS updates), the mobility model, the energy consumption model, DG tree update policy and channel access policies as well as define the structural metrics and performance metrics.

Simulation Parameters: The network dimensions is 100m x 100m (Area A = 10,000 m2) and the sink is assumed to be outside the network: at (50, 300). The number of nodes (N) in the network is set to be 50 and 100, and the transmission range (R) per node values used are 25m and 35m. The average number of neighbors per node is computed using the formula: $\pi R^2 N/A$. Accordingly, we have the following scenarios of network density: low density (N = 50, R = 25m, Avg. # neighbors per node = 9.8), low-moderate density (N = 50, R = 35m, Avg. # neighbors per node = 19.2), moderate-high density (N = 100, R = 25m, Avg. # neighbors per node = 19.6) and high density (N = 100, R = 35m, Avg. # neighbors per node = 38.5). The maximum velocity of a node (vmax) is set to be: 1 m/s (low mobility), 3 m/s (low-moderate mobility), 5 m/s (moderate-high mobility) and 10 m/s (high mobility). Thus, we have a total of sixteen scenarios of various combinations of network density and node mobility. We generated 100 instances of node mobility profiles for each of the above sixteen scenarios of network density and node mobility and averaged the results (with respect to the performance metrics and structural metrics discussed below) obtained for the MaxBLW-DG algorithm incorporated with the BPI and LET-based link selection strategies run on these 100 instances.

Mobility Model: To start with, the nodes are uniform-randomly distributed throughout the network. Mobility of the nodes is modeled according to the Random Waypoint model [3] with the nodes moving continuously (zero pause time) and independent of each other. A node decides to move from its current location to a randomly chosen location within the network with a velocity uniform-randomly chosen from [0...vmax]; after reaching the chosen location, the node continues its movement by randomly choosing another location with a different randomly chosen velocity from the above range. A node continues its movement like this throughout the simulation. We record the instances of direction change and the corresponding location and velocity to construct (offline) a mobility profile for each node and feed in this mobility profile to the MaxBLW-DG tree algorithm.

Energy Consumption Model: Nodes are assumed to be of sufficient energy so that there are no node failures due to exhaustion of energy. The energy consumed at a node for data aggregation is the sum of the energy lost in receiving the aggregated data from each of its child nodes, fusing its own data with that of the aggregated data and transmitting the final aggregated data to its upstream node in the DG tree. The energy consumed at a node for broadcast tree discovery is the sum of the energy lost to receive the broadcast control message from each of its neighbors and to the transmit the control message in its neighborhood, if the conditions for rebroadcast are met. The energy consumption model used is a first-order radio model [28] that has been used in several of the previous work [7, 11] in the literature. According to this model: (i) the energy consumed at a sensor node to transmit a k-bit message over a distance d is given by: ETX(k, d) = Eelec*k + $\in_{amp}$ *k*d2, where Eelec = 50 nJ/bit is the energy lost to run the radio transmitter or receiver circuitry and $\in_{amp}$ = 100 pJ/bit/m2 is the energy lost to run the transmitter amplifier; (ii) the energy lost at a sensor node to broadcast a k-bit message to all its neighbors within the transmission range R is simply given by ETX(k, R); the energy consumed at a sensor node to receive a k-bit message is ERX(k) = Eelec *k. The total energy consumed at a sensor node to receive k-bit broadcast messages transmitted by all of its n-neighbors is simply given by n * ERX(k). We do not take into consideration the energy lost due to periodic beacon exchange as both the LET and BPI-based link selection strategies considered in this research use it to determine the link weights.

Data Size and Frequency of LSS Updates: We conduct the simulations for 2000 rounds (one round for every 0.25 seconds: a total of 500 seconds). The LSS scores of the links are estimated in the neighborhood of the nodes for every second. For each round: data gets aggregated across the network, starting from the leaf nodes and proceeding all the way to the LEADER node of the DG tree; the LEADER node forwards the final aggregated data to the sink. The data size is assumed to remain the same during network-wide aggregation. That is, the size of the aggregated data is assumed to be the same as the size of the data collected at the individual sensor nodes. The data size is 2000 bits and the size of the control messages used for tree configuration and maintenance is assumed to be 400 bits (sufficiently large enough to accommodate the various fields in the control messages).

DG Tree Update Policy: Every time a DG tree is needed, the sink collects the weights of the links of the sensor nodes using a network-wide broadcast. The node with the largest sum of the link weights is considered as the root node (a.k.a. LEADER node) and the sink node sends a control message to the LEADER node to initiate tree discovery (a process also called tree

reconfiguration). A DG tree is used as long as it exists: this is referred to as the Least Overhead Routing Approach (LORA) [1] in the literature of mobile ad hoc networks.

Channel Access Policy: Note that in a particular timeslot, an intermediate node could collect data from only one of its child nodes (using Time Division Multiple Access, TDMA [35]) if the latter has its aggregated data available, and an intermediate node could transmit upstream its aggregated data only after receiving the same from each of its child nodes and aggregating with its own. An intermediate node could collect data from one of its child nodes at the same time (using Code Division Multiple Access, CDMA [35]) as any other intermediate node collects data from any of its child nodes. We assume that sufficient number of CDMA and TDMA codes are available at the sensor nodes (as needed) to facilitate data aggregation in the minimum number of time slots.

Structural Metrics: We evaluated the following three structural metrics: (S-i) Tree Height, TH: The tree height is the maximum of the level numbers of the vertices (i.e., the number of hops) from the root node of the DG tree (with the root node considered to be at level 0). (S-ii) Fraction of Leaf Nodes, FLN: The fraction of leaf nodes is the ratio of the number of leaf nodes to the total number of nodes in the network graph. (S-iii) Average Number of Child Nodes per Intermediate Node, CNI: The average number of child nodes per intermediate node is the weighted average of the number of child nodes per intermediate node considered across all intermediate nodes.

Performance Metrics: We evaluated the following three performance metrics: (P-i) Tree Lifetime, TL: The tree lifetime is the number of rounds a DG tree exists before one or more of its links fail due to node mobility, averaged over the duration of a simulation session. (P-ii) Aggregation Delay per Round, ADR: The aggregation delay per round is the minimum number of timeslots (computed as per algorithm [25]) it takes for data to get aggregated along the edges of the DG tree and reach the root node, averaged across all the rounds. (P-iii) Energy Consumption per Round, ECR: The energy consumed per round is the sum of the energy consumed at each of the nodes for data aggregation in the network plus the energy lost due to broadcast tree discoveries if the DG tree was reconfigured at the beginning of the round. We average the energy consumed across all the rounds of a simulation session.

## 5.2 Normalized comprehensive relative performance (NCRP) score

As described in Section 5.4, we observe a complex tradeoff between the three performance metrics: tree lifetime, energy consumption per round and aggregation delay per round. Since the performance metrics incur different levels of magnitude, we propose to bring the values incurred for these metrics on a common scale of 0 to 1 using the method of normalization and propose to prefer the link selection strategy that incurs the largest value for the normalized score (or the complement of the normalized score, as appropriate) with respect to the individual metrics and/or with respect to the normalized comprehensive relative performance (NCRP) score (introduced below). In other words, the idea is to normalize the values incurred for each of the performance metrics incurred for the BPI and LET link selection strategies for a particular simulation scenario and compute a normalized comprehensive relative performance (NCRP) score with respect to the performance metrics (as shown below).

As we seek for a larger tree lifetime (see Section 5.4), lower energy consumption per round (see Section 5.5) and lower aggregation delay per round (see Section 5.6), we use the normalized values for the tree lifetime (*TL*), but complement of the normalized values for the energy consumption per round (*ECR*) and aggregation delay per round (*ADR*) to compute the NCRP score as a weighted average (weight = 1/3 for each metric) of these three values.

Complement of Norm. *ECR* = 1 - Normalized *ECR*

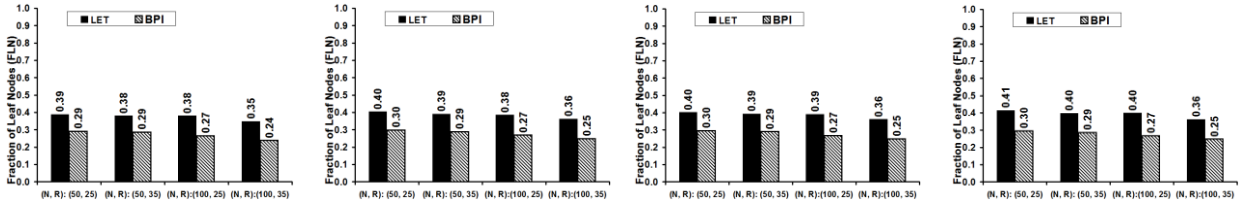
Complement of Norm. *ADR* = 1 - Normalized *ADR*

*NCRP* = {Normalized *TL* + Complement of Norm. *ECR* + Complement of Norm. *ADR*}/3 .....(3)

## 5.3 Structural metrics

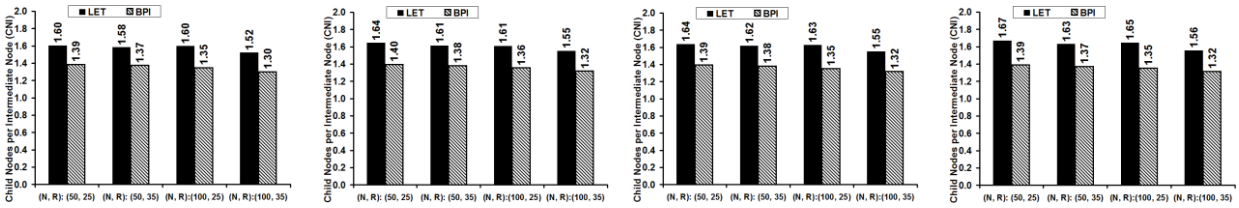
In this section, we illustrate the results obtained with respect to the structural metrics for the DG trees determined based on the BPI and LET strategies. For lower energy consumption and lower aggregation delay per round, we would desire to have DG trees with a lower number of child nodes per intermediate node (so that an intermediate node can spend less energy in receiving data from each of its child nodes as well as aggregate data from its child nodes in fewer time slots) and at the same time a larger fraction of leaf nodes (so that the energy lost due to receptions could be lower and the number of nodes that readily have the data to transmit could be larger). However, we observe that it would not be possible to simultaneously maximize the fraction of leaf nodes as well as minimize the number of child nodes per intermediate node. As the fraction of leaf nodes in a DG tree increases, the fraction of intermediate nodes in the DG tree is bound to decrease and hence the number of child nodes per intermediate node is bound to only increase. We also observe a similar trend in the results (see Figures 5-6) for the structural metrics obtained for the DG trees based the LET and BPI strategies.

The LET-based DG trees incur a larger fraction of leaf nodes (desirable for lower energy consumption and lower aggregation delay per round) and lower height (desirable for lower aggregation delay per round), but also simultaneously incur a larger number of child nodes per intermediate node. The BPI-based DG trees incur a relatively lower number of child nodes per intermediate node (desirable for lower energy consumption and lower aggregation delay per round), but also incur a lower fraction of leaf nodes. Thus, as envisioned previously, we observe a tradeoff between the fraction of leaf nodes and the number of child nodes per intermediate node.



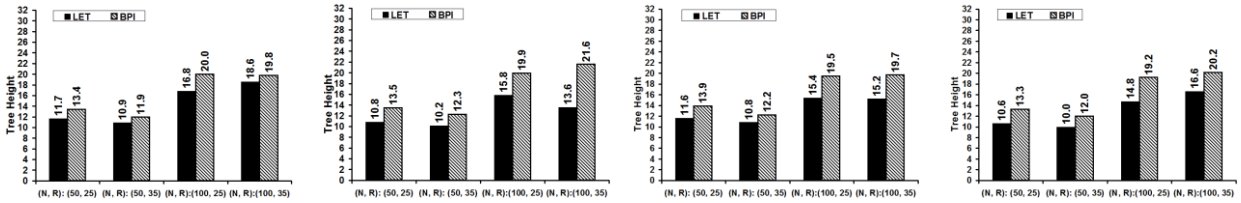

(a) $v_{max}$ = 1 m/s (b) $v_{max}$ = 3 m/s (c) $v_{max}$ = 5 m/s (d) $v_{max}$ = 10 m/s

Figure 5: Average Fraction of Leaf Nodes.




(a) $v_{max}$ = 1 m/s (b) $v_{max}$ = 3 m/s (c) $v_{max}$ = 5 m/s (d) $v_{max}$ = 10 m/s

Figure 6: Average Number of Child Nodes per Intermediate Node.



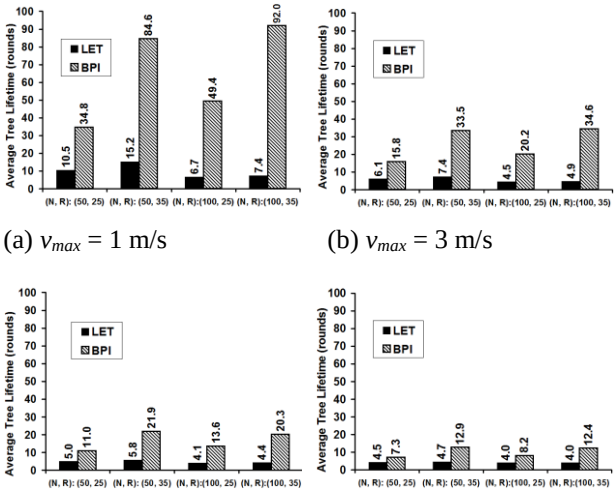

(a) $v_{max}$ = 1 m/s (b) $v_{max}$ = 3 m/s (c) $v_{max}$ = 5 m/s (d) $v_{max}$ = 10 m/s

Figure 7: Average Tree Height.

Since the structural metrics are not dependent on node mobility, for a given network density: we observe the values incurred for the structural metrics to be independent of node mobility. For a given level of node mobility, we observe the fraction of leaf nodes as well as the average number of child nodes per intermediate node (incurred for both the LET and BPI-based DG trees) to decrease with increase in network density. On the other hand, for a given level of node mobility, we observe the tree height to increase with increase in network density (especially, as we increase from 50 to 100 nodes).

## 5.4 Tree lifetime

The BPI-DG trees incur significantly larger values for the tree lifetime (see Figure 8) compared to that of the LET-DG trees. The lifetime of the BPI-DG trees could be as large as 12 times the lifetime of the LET-DG trees (especially in scenarios of high network density and low node mobility). Even in the worst case (scenarios of low network density and high node mobility), the lifetime of the BPI-DG trees is at least 60% larger than the lifetime of the LET-DG trees. When considered across all the 16 scenarios of network density and node mobility (refer Figure 9), the normalized values (with respect to tree lifetime) for the BPI-DG trees is at least 0.85; whereas, the normalized values for the LET-DG trees is at most 0.52.



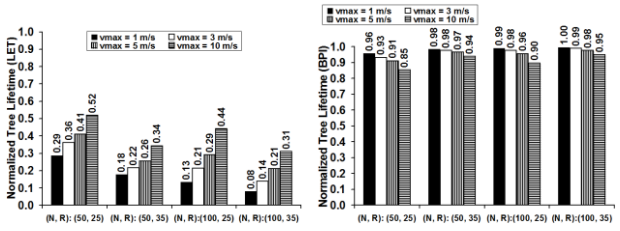

(a) $v_{max}$ = 1 m/s (b) $v_{max}$ = 3 m/s

(c) $v_{max}$ = 5 m/s (d) $v_{max}$ = 10 m/s

Figure 8: Absolute Value of Average Tree Lifetime.

From Figure 8, we could observe that for a fixed level of node mobility: the average lifetimes for the BPI-DG trees relatively increase with increase in network density, whereas the average lifetimes for the LET-DG trees relatively decrease with increase in network density. From Figure 9, for a given level of network density: we could observe that the relative performance

of the LET-DG trees with respect to tree lifetime improves with increase in node mobility. On the other hand, the relative performance of the BPI-DG trees with respect to tree lifetime remains almost the same or only marginally degrades with increase in node mobility. Thus, with respect to tree lifetime, the BPI-DG trees are relatively more scalable (i.e., are robust to increase in network density for a given level of node mobility) and remains relatively about the same (with increase in node mobility for a given network density). Such observations on the relative performance of the link selection strategies cannot be easily assessed by simply looking at the actual values incurred for tree lifetime in Figure 8 (or for that matter any other performance metric).




(a) LET (b) BPI

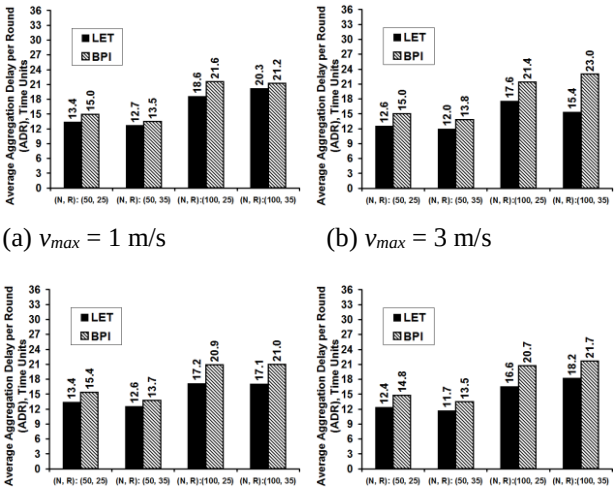
Figure 9: Normalized Value of Average Tree Lifetime.

## 5.5 Aggregation delay per round

From Figures 10-11, we observe the LET-DG trees to incur lower *ADR* values for all conditions of network density and node mobility. For a given level of node mobility, the difference in the magnitude of the *ADR* values between the BPI-DG trees and the LET-DG trees increases with increase in network density. For a given network density, the *ADR* values incurred for the DG trees based on a particular link selection strategy remain about the same (there is no particular or a significant trend of variation) at different levels of node mobility.

As we prefer a link selection strategy to yield lower aggregation delay per round for the DG trees, we plot the complement of the normalized *ADR* values (instead of just the normalized *ADR* values) of Figure 10 in Figure 11. The *ADR* values (shown in Figure 10) incurred for both the LET-DG and BPI-DG trees appear to be directly proportional and positively correlated with the height of the DG trees (shown in Figure 7). Though the absolute *ADR* values (Figure 10) are observed to increase with increase in network density for a given level of node mobility, there is no change in the trend of the normalized *ADR* values (a measure of the relative performance) incurred for the DG trees based on both LET and BPI (for a given level of node mobility, the complement of the normalized *ADR* values for either LET or BPI almost remains the same with increase in network density).

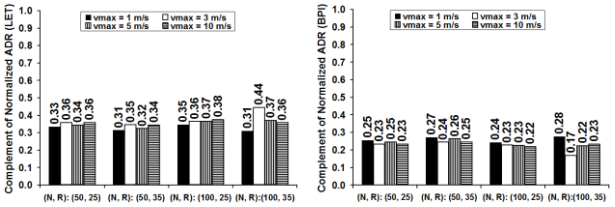
Unlike the exceptionally high values for the tree lifetime incurred with the BPI-DG trees and relatively very poor tree lifetime (see Figures 8-9) observed for the LET-DG trees, (the complement of) the normalized *ADR* values incurred for the DG trees determined based on LET and BPI are not far different. In the case of tree lifetime, the difference in the normalized values for the lifetime of the LET and BPI- based DG trees is at least 0.33 and is as large as 0.92 (with a high median of 0.69). On the other hand, in the case of aggregation delay per round, the difference in the complement of the normalized ADR values of the LET and BPI-based DG trees is at most 0.27 (with a low median of 0.11).




(a) $v_{max}$ = 1 m/s (b) $v_{max}$ = 3 m/s

(c) $v_{max}$ = 5 m/s (d) $v_{max}$ = 10 m/s

Figure 10: Absolute Value of Average Aggregation Delay per Round (in time units).



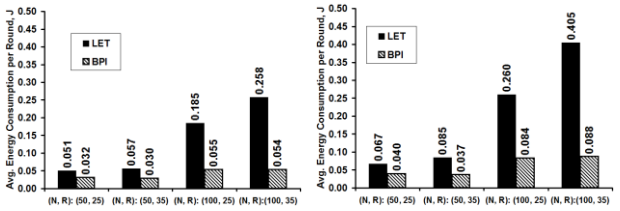

(a) LET (b) BPI

Figure 11: Complement of the Normalized Value of Average Aggregation Delay per Round.

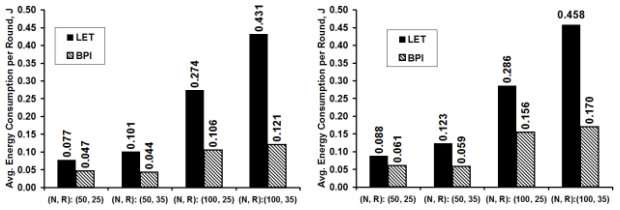
## 5.6 Energy consumption per round

The BPI-DG trees incur lower values for energy consumption per round (*ECR*) for all scenarios of network density and node mobility (see Figures 12-13), and the LET-DG trees incur larger *ECR* values for all scenarios. The relatively better performance of the BPI-based DG trees with respect to *ECR* could be primarily attributed to the less frequent network-wide broadcasts (due to a larger tree lifetime) and the short distance nature of the links that are part of the transmissions during data aggregation. For a given level of node mobility: the difference in the *ECR* values between the BPI and LET-based DG trees increases with increase in network density (attributed to the relatively unstable LET-based DG trees with increase in network density). Of course, for a fixed network density, the difference in the *ECR* values increase with increase in node mobility. Though both LET and BPI incur an increase in the magnitude for the *ECR* values with increase in network

density and/or node mobility, the *ECR* values for the LET-DG trees are significantly larger than those incurred for the BPI-DG trees.



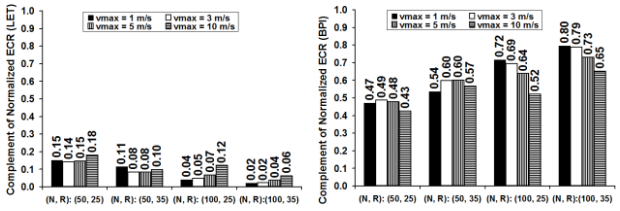

(a) $v_{max}$ = 1 m/s (b) $v_{max}$ = 3 m/s




(c) $v_{max}$ = 5 m/s (d) $v_{max}$ = 10 m/s

Figure 12: Absolute Value of Average Energy Consumption per Round (in Joules).

For a given level of node mobility (see Figure 13): the complement of the normalized *ECR* values for the BPI-DG trees increases with increase in network density; whereas, the complement of the normalized *ECR* values for the LET-DG trees decreases with increase in network density. Thus, for a given level of node mobility: the relative performance (with respect to *ECR*) of the BPI-DG trees vis-a-vis the LET-DG trees improves with increase in network density. On the other hand, (see Figure 13), for a given network density: the complement of the normalized *ECR* values for the LET-DG trees slightly increase with increase in node mobility; whereas, the complement of the normalized *ECR* values for the BPI-DG trees slightly decrease with increase in node mobility, more visibly in networks of high density.

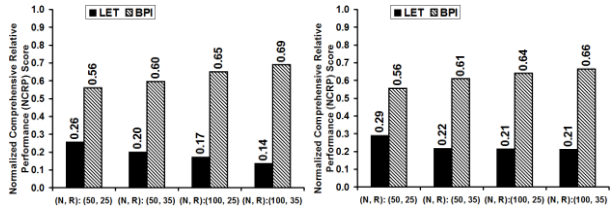



(a) LET (b) BPI

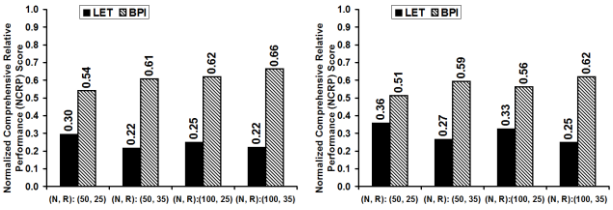
Figure 13: Complement of the Normalized Value of Average Energy Consumption per Round.

## 5.7 Analysis of the relative performance tradeoff based on the NCRP scores

We observe the lifetime incurred for the BPI-DG trees to be significantly larger than that of the LET-DG trees. Likewise, the average energy consumption per round incurred for the BPI-DG trees is lower than that incurred for the LET-DG trees. On the other hand, the aggregation delay per round incurred for the LET-DG trees is lower than the aggregation delay per round values incurred for the BPI-DG trees. This illustrates a complex {tree lifetime, energy consumption per round} vs. {aggregation delay per round} tradeoff. We use the normalization approach (introduced in Section 5.2) to analyze this tradeoff with respect to the above three performance metrics between LET and BPI. We observe (from Figure 14) the BPI-based link selection strategy to effectively balance this tradeoff and incur larger values for the NCRP score under all the 16 different scenarios of network density and node mobility. A closer look at the actual values (from Figures 8, 10 and 12) and normalized values (from Figures 9, 11 and 13) for the three individual performance metrics illustrates the same.




(a) $v_{max}$ = 1 m/s (b) $v_{max}$ = 3 m/s




(c) $v_{max}$ = 5 m/s (d) $v_{max}$ = 10 m/s

Figure 14: Normalized Comprehensive Relative Performance Score for the Link Selection Strategies.

With respect to the impact of network density and node mobility: the NCRP scores for the LET-DG trees decrease with increase in network density (for a fixed level of node mobility) and increase with increase in node mobility (for a fixed network density). On the other hand, the NCRP scores for the BPI-DG trees increase with increase in network density (for a fixed level of node mobility) and remain about the same with increase in node mobility (for a fixed network density). Thus, we observe the overall relative performance of the BPI-DG trees to only improve (or remain the same) with increase in network density and/or node mobility.

Table 1 provides a comprehensive overview of the simulation results, identifying the link selection strategy that yields the most desirable values for the structural metrics and performance metrics. With respect to the structural metrics, we desire to have larger values for the fraction of leaf nodes and lower values for the number of child nodes per intermediate node and tree height. With respect to the performance metrics, we desire to have larger values for tree lifetime and lower values for the energy consumption per round and aggregation delay per round. With respect to the NCRP score (see equation 3 for the formulation), we desire to have values closer to 1. On these lines, the LET strategy returns the most desirable values for two of the three structural metrics as

Table 1: Link Selection Strategies (LET vs. BPI) that Incur the most Desirable Values for the Structural Metrics and Performance Metrics as well as the NCRP Score.

| # nodes | Tr. range | $v_{max}$ | *FLN* | *CNI* | *TH* | *TL* | *ECR* | *ADR* | *NCRP* |
|---|---|---|---|---|---|---|---|---|---|
| 50 | 25m | 1 m/s | LET | BPI | LET | BPI | BPI | LET | BPI |
| | | 3 m/s | LET | BPI | LET | BPI | BPI | LET | BPI |
| | | 5 m/s | LET | BPI | LET | BPI | BPI | LET | BPI |
| | | 10 m/s | LET | BPI | LET | BPI | BPI | LET | BPI |
| 50 | 35m | 1 m/s | LET | BPI | LET | BPI | BPI | LET | BPI |
| | | 3 m/s | LET | BPI | LET | BPI | BPI | LET | BPI |
| | | 5 m/s | LET | BPI | LET | BPI | BPI | LET | BPI |
| | | 10 m/s | LET | BPI | LET | BPI | BPI | LET | BPI |
| 100 | 25m | 1 m/s | LET | BPI | LET | BPI | BPI | LET | BPI |
| | | 3 m/s | LET | BPI | LET | BPI | BPI | LET | BPI |
| | | 5 m/s | LET | BPI | LET | BPI | BPI | LET | BPI |
| | | 10 m/s | LET | BPI | LET | BPI | BPI | LET | BPI |
| 100 | 35m | 1 m/s | LET | BPI | LET | BPI | BPI | LET | BPI |
| | | 3 m/s | LET | BPI | LET | BPI | BPI | LET | BPI |
| | | 5 m/s | LET | BPI | LET | BPI | BPI | LET | BPI |
| | | 10 m/s | LET | BPI | LET | BPI | BPI | LET | BPI |

well as the aggregation delay per round (all of which are not dependent on node mobility); however, the BPI strategy is useful to discover stable DG trees (larger tree lifetime) that also incur a lower energy consumption per round (attributed to the less frequent network-wide broadcasts and short distance nature of the links). The relatively better performance of the BPI-DG trees with respect to the tree lifetime and energy consumption per round and competitive values for the aggregation delay per round lift the normalized comprehensive relative performance (NCRP) scores to be above that of the LET strategy. The minimum and maximum difference in the NCRP scores incurred for the BPI-DG trees vis-a-vis the LET-DG trees are respectively 0.15 (observed in scenario of low network density and high node mobility) and 0.55 (observed in scenario of high network density and low node mobility). The median difference in the NCRP scores is 0.38.

# 6 Conclusions

The high-level contribution of this paper is a proposal to use the Bipartivity Index (BPI) metric (a spectral graph-theoretic metric used in complex network analysis) to determine stable data gathering (DG) trees for mobile sensor networks (MSNs). Our hypothesis in this research is that the end nodes of short distance links (the Euclidean distance between the end nodes of the link is far less than the transmission range of the nodes) are more likely to share a significant fraction of their neighborhood (and vice-versa). As short distance links are more likely to be stable too (and vice-versa), we propose to use the BPI strategy to evaluate and quantify the extent of shared neighborhood of the end vertices of the edges for determining stable DG trees in mobile sensor networks.

We model the neighborhood of the end vertices of an edge *u-v* as an *egocentric network* $EG_{u\text{-}v}$ comprising of the end vertices and their neighbors as *nodes* and the edges incident on the end vertices as *links*. We have shown (through detailed theoretical analysis and illustrative examples) that edges whose egocentric networks have smaller values for the BPI are more likely to be short distance links. The egocentric network for an edge and its BPI score could be independently determined by the two end vertices of the edge based on just the one-hop neighborhood information and without knowledge about the location and mobility of the nodes. We quantify the link stability score (LSS) for an edge *u-v* as 1 - BPI($EG_{u\text{-}v}$). We define the bottleneck link weight of a path as the minimum of the weights of the constituent links on the path. Whenever a DG tree is required, we determine the maximum bottleneck link weight-based DG tree for which the bottleneck link weight of the path from any node to the root node is the maximum (the root node is the node with the largest sum of the LSS scores of its incident links).

We have compared the performance of the BPI-based DG trees with that of the DG trees determined based on the predicted link expiration time (LET) - the only well-known strategy so far [19] to determine stable DG trees for MSNs. We observe the BPI-DG trees to be significantly more stable as well as incur a lower energy consumption per round compared to that of the LET-DG trees. On the other hand, we observe the aggregation delay per round incurred for the LET-DG trees to be lower than the aggregation delay per round incurred for the BPI-DG trees. We thus observe a complex {tree lifetime, energy consumption per round} vs. {aggregation delay per round} tradeoff. We attribute this tradeoff to the unstable nature of the LET-DG trees (leading to more energy-intensive network-wide broadcast tree discoveries) and lower height as well as a larger fraction of leaf nodes (contributing to a lower aggregation delay per round).

Finally, we propose the use of a normalization-based approach to evaluate the relative performance of the link selection strategies in a scale of 0...1 and thereby

overcome the difficulty arising in analyzing the tradeoffs among the performance metrics whose values fall under different levels of magnitude (as is the case for the three performance metrics studied in this paper). We illustrate the use of the normalization-based approach to identify the link selection strategy that best balances the performance tradeoff as well as whose relative performance is more scalable (with increase in network density and/or node mobility). We observe the BPI-based link selection strategy to yield DG trees that incur the largest values for the normalized comprehensive relative performance (NCRP) scores under all the 16 scenarios of network density and node mobility. To vindicate the larger NCRP scores, we observe the BPI-based DG trees to simultaneously incur larger values for tree lifetime and lower values for energy consumption per round and not so relatively high values for aggregation delay per round. We observe the comprehensive relative performance of the BPI-DG trees to be more scalable with increase in network density and not much affected with increase in node mobility.

# 7 Acknowledgment

The work leading to this paper is funded through the Minority Leaders Program, "A Stable Trustworthy Neighborhood Scheme for Secure Mobile Sensor Networks," funded by the Clarkson Aerospace/US Air Force Research Lab. Subcontract Number: JACK 15-S7700-02-C2. The AFRL public clearance approval number for this article is 88ABW-2017-0475. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the funding agency.

# 8 References

[1] M. Abolhasan, T. Wysocki, E. Dutkiewicz, "A Review of Routing Protocols for Mobile Ad hoc Networks," *Ad hoc Networks*, vol. 2, no. 1, pp. 1-22, 2004.

[2] T. Banerjee, B. Xie, J. H. Jun and D. P. Agarwal, "LIMOC: Enhancing the Lifetime of a Sensor Network with Mobile Clusterheads," *Proceedings of the Vehicular Technology Conference Fall*, pp. 133-137, Baltimore, MD, USA, September 30 - October 3, 2007.

[3] C. Bettstetter, H. Hartenstein and X. Perez-Costa, "Stochastic Properties of the Random-Way Point Mobility Model," *Wireless Networks*, vol. 10, no. 5, pp. 555-567, September 2004.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms*, 3rd Edition, MIT Press, July 2009.

[5] S. Deng, J. Li and L. Shen, "Mobility-based Clustering Protocol for Wireless Sensor Networks with Mobile Nodes," *IET Wireless Sensor Systems*, vol. 1, no. 1, pp. 39-47, March 2011.

[6] E. Estrada and J. A. Rodriguez-Velazquez, "Spectral Measures of Bipartivity in Complex Networks," *Physical Review E* 72, 046105, pp. 1-6, 2005.

[7] W. Heinzelman, A. Chandrakasan and H. Balakarishnan, "Energy-Efficient Communication Protocols for Wireless Microsensor Networks," *Proceedings of the Hawaaian International Conference on Systems Science*, Maui, HI, USA, January 2000.

[8] B. Hofmann-Wellenhof, H. Lichtenegger and J. Collins, *Global Positioning System*: *Theory and Practice*, 5th Edition, Springer, October 2013.

[9] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan and S. Madden, "CarTel: A Distributed Mobile Sensor Computing System," *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, pp. 125-138, Boulder, CO, USA, November 2006.

[10] Y. Lai, J. Xie, Z. Lin, T. Wang and M. Liao, "Adaptive Data Gathering in Mobile Sensor Networks using Speedy Mobile Elements," *Sensors*, vol. 15, no. 9, pp. 23218-23248, 2015.

[11] S. Lindsey, C. Raghavendra and K. M. Sivalingam, "Data Gathering Algorithms in Sensor Networks using Energy Metrics," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 9, pp. 924-935, September 2002.

[12] C-M. Liu, C-H. Lee and L-C. Wang, "Distributed Clustering Algorithms for Data Gathering in Wireless Mobile Sensor Networks," *Journal of Parallel and Distributed Computing*, vol. 67, no. 11, pp. 1187-1200, November 2007.

[13] P. V. Marsden, "Egocentric and Sociocentric Measures of Network Centrality," vol. 24, no. 4, pp. 407-422, October 2002.

[14] M. Ma and Y. Yang, "SenCar: An Energy-Efficient Data Gathering Mechanism for Large-Scale Multihop Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 10, pp. 1476-1488, October 2007.

[15] M. Macuha, M. Tariq and T. Sato, "Data Collection Method for Mobile Sensor Networks Based on the Theory of Thermal Fields," *Sensors*, vol. 11, no. 7, pp. 7188-7203, July 2011.

[16] N. Meghanathan, "A Data Gathering Algorithm based on Energy-aware Connected Dominating Sets to Minimize Energy Consumption and Maximize Node Lifetime in Wireless Sensor Networks," *International Journal of Interdisciplinary Telecommunications and Networking*, vol. 2, no. 3, pp. 1-17, July-September 2010.

[17] N. Meghanathan, "Exploring the Performance Tradeoffs among Stability-Oriented Routing Protocols for Mobile Ad hoc Networks," *Network Protocols and Algorithms – Special Issue on Data Dissemination for Large scale Complex Critical Infrastructures*, vol. 2, no. 3, pp. 18-36, November 2010.

[18] N. Meghanathan, "A Comprehensive Review and Performance Analysis of Data Gathering Algorithms for Wireless Sensor Networks,"

*International Journal of Interdisciplinary Telecommunications and Networking*, vol. 4, no. 2, pp. 1-29, April-June 2012.

[19] N. Meghanathan, "Link Expiration Time and Minimum Distance Spanning Trees based Distributed Data Gathering Algorithms for Wireless Mobile Sensor Networks," *International Journal of Communication Networks and Information Security*, vol. 4, no. 3, pp. 196-206, December 2012.

[20] N. Meghanathan, "Routing Protocols to Determine Stable Paths and Trees using the Inverse of Predicted Link Expiration times for Mobile Ad hoc Networks," *International Journal of Mobile Network Design and Innovation*, vol. 4, no. 4, pp. 214-234, June 2012.

[21] N. Meghanathan and P. Mumford, "A Benchmarking Algorithm to Determine the Sequence of Stable Data Gathering Trees for Wireless Mobile Sensor Networks," *Informatica – An International Journal of Computing and Informatics*, vol. 37, no. 3, pp. 315-338, October 2013.

[22] N. Meghanathan, *Recent Advances in Ad Hoc Networks Research*, Nova Science Publishers, August 2014.

[23] N. Meghanathan, "Stability-based and Energy-Efficient Distributed Data Gathering Algorithms for Mobile Sensor Networks," *Ad hoc Networks*, vol. 19, pp. 111-131, August 2014.

[24] N. Meghanathan, "A Generic Algorithm to Determine Maximum Bottleneck Node Weight-based Data Gathering Trees for Wireless Sensor Networks," *Network Protocols and Algorithms*, vol. 7, no. 3, pp. 18-51, November 2015.

[25] N. Meghanathan, "A Benchmarking Algorithm to Determine Minimum Aggregation Delay for Data Gathering Trees and an Analysis of the Diameter-Aggregation Delay Tradeoff," *Algorithms*, vol. 8, no. 3, pp. 435-458, July 2015.

[26] N. Meghanathan, "A Greedy Algorithm for Neighborhood Overlap-based Community Detection," *Algorithms*, vol. 9, no. 1, p. 8: 1-26, 2016.

[27] P. De Meo, E. Ferrara, G. Fiumara and A. Provetti, "On Facebook, Most Ties are Weak," *Communications of the ACM*, vol. 57, no. 11, pp. 78-84, November 2014.

[28] T. S. Rappaport, *Wireless Communications: Principles and Practice*, 2nd edition, Prentice Hall, January 2002.

[29] G. Santhosh Kumar, M. V. Vinu Paul and K. Jacob Poulose, "Mobility Metric based LEACH-Mobile Protocol," *Proceedings of the 16th International Conference on Advanced Computing and Communications*, pp. 248-253, Chennai, India, December 2008.

[30] H. K. D. Sarma, R. Mall and A. Kar, "$E^2R^2$: Energy-Efficient and Reliable Routing for Mobile Wireless Sensor Networks," *IEEE Systems Journal*, vol. 10, no. 2, pp. 604-616, April 2015.

[31] W. Su and M. Gerla, "IPv6 Flow Handoff in Ad hoc Wireless Networks using Mobility Prediction," *Proceedings of the IEEE Global Telecommunications Conference*, pp. 271-275, December 1999.

[32] D. Tao, S. Tang and H. Ma, "Low Cost Data Gathering using Mobile Hybrid Sensor Networks," *Lecture Notes in Computer Science*, vol. 7363, pp. 193-206, July 2012.

[33] Y-C. Tseng, F-J. Wu and W-T. Lai, "Opportunistic Data Collection for Disconnected Wireless Sensor Networks by Mobile Mules," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1150-1164, May 2013.

[34] R. Velmani and B. Kaarthick, "An Energy Efficient Data Gathering in Dense Mobile Wireless Sensor Networks," *International Scholarly Research Notices Sensor Networks*, vol. 2014, Article ID: 518268, 10 pages, 2014.

[35] A. J. Viterbi, *CDMA*: *Principles of Spread Spectrum Communication*, 1st Edition, Prentice Hall, April 1995.

[36] H. Zhang and J. C. Hou, "Maintaining Sensing Coverage and Connectivity in Large Sensor Networks," *Wireless Ad hoc and Sensor Networks*: *An International Journal*, vol. 1, no. 1-2, pp. 89-123, January 2005.