DV-Hop Localization Algorithms in Wireless Sensor Networks

by

Xiaoxi Du
B.Eng., Dalian University of Technology, 2012

A Project Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF ENGINEERING

in the Department of Electrical & Computer Engineering

DV-Hop Localization Algorithms in Wireless Sensor Networks

by

Xiaoxi Du
B.Eng., Dalian University of Technology, 2012

Supervisory Committee

Dr. T. Aaron Gulliver, Supervisor
(Department of Electrical & Computer Engineering As Candidate)

Dr. Wu-Sheng Lu, Departmental Member
(Department of Electrical & Computer Engineering As Candidate)

Supervisory Committee

Dr. T. Aaron Gulliver, Supervisor
(Department of Electrical & Computer Engineering As Candidate)

Dr. Wu-Sheng Lu, Departmental Member
(Department of Electrical & Computer Engineering As Candidate)

ABSTRACT

Localization algorithms are becoming more and more important in wireless sensor networks (WSNs). Localization algorithms can be classified into two categories: range-based and range-free. Distance vector-hop (DV-Hop) is a range-free localization algorithm using the numbers of hops to estimate distance. This project focuses on the performance of DV-Hop and two variations of DV-Hop: DV-Hop with correction (CDV-Hop) and improved DV-Hop (IDV-Hop). The simulation of these algorithms is written in the Java programming language using an open source WSN simulator called NetTopo. This project investigates the performance of these algorithms with multiple factors: the anchor node density (AND), the number of sensor nodes (SNN), the transmission radius (TR) of the sensor nodes, and the neighborhood size (NS) of the sensor nodes. The connected k-neighborhood (CKN) algorithm is used as a tool in the investigation of the NS. CKN extracts a subset of sensor nodes with k-neighborhood connectivity (k approximates the NS of the network of the extracted subset of sensor nodes). Based on the above factors, the localization accuracy of the three DV-Hop algorithms is examined. Experimental results compare the performance of the three algorithms and give hints on the choice of the parameters.

Contents

List of Tables

List of Figures

Acronyms

AND anchor node density

AoA angle of arrival

APIT approximate point-in triangulation test

CDV-Hop DV-Hop with correction

CKN connected k-neighborhood

DPH distance per hop

DV-Hop distance vector-hop

GPS global positioning system

GUI graphical user interface

HC hop count

HT hop table

IDV-Hop improved DV-Hop

LS least squares

NS neighborhood size

RSSI received signal strength indicator

SNN number of sensor nodes

TDoA time difference of arrival

ToA time of arrival

TR transmission radius

WSN wireless sensor network

ACKNOWLEDGEMENTS

I would like to thank:

Dr. T. Aaron Gulliver, for mentoring, support, encouragement, and patience.

All life demands struggle. Those who have everything given to them become lazy, selfish, and insensitive to the real values of life. The very striving and hard work that we so constantly try to avoid is the major building block in the person we are today.
Pope Paul VI

Chapter 1

Introduction

1.1 Wireless sensor networks and localization algorithms

A wireless sensor network (WSN) refers to a network of spatially distributed wireless sensor nodes used to monitor environmental or physical conditions. With the recent development of sensor devices, radio techniques, and low power electronics, WSNs have been applied in a wide range of areas, including military surveillance, environmental monitoring and indoor tracking [1]. Many applications rely on knowing the locations of the sensor nodes. Therefore, research in localization algorithms for WSNs has become an important topic [2].

The sensor nodes in WSNs can be divided into two types: anchor nodes and unknown nodes. The locations of anchor nodes are acquired by global positioning system (GPS) or manual configuration. Unknown nodes obtain their estimated locations via localization algorithms.

1.2 Classification of localization algorithms

Localization algorithms for WSNs can be classified into two categories: range-based and range-free. Range-based schemes use range measurements between sensor nodes to estimate the coordinates of unknown nodes, whereas range-free approaches do not need this information.

1.2.1 Range-based localization algorithms

Several range-based techniques have been proposed for WSNs, such as time of arrival (ToA) [3], time difference of arrival (TDoA) [4], angle of arrival (AoA) [5] and received signal strength indicator (RSSI) [6]. In ToA [3], each sensor node sends a signal to its neighbors, and uses signal transmission times to estimate the distances to these neighbors.

TDoA [4] uses two signals with different speeds of propagation, and does not need time synchronization. Usually, the signals are chosen as radio and ultrasound. Each sensor node first sends a radio signal to its neighbors, and sends an ultrasound signal with a fixed pattern of chirps [7] after a fixed interval of time. Once a neighbor receives the radio signal, it records the time, and records another time once the ultrasound is detected. Based on the two times recorded and the fixed time interval, the sensor node can calculate the distance to this neighbor.

In AoA [5], each receiving sensor node is equipped with multiple spatially separated antennas or ultrasound receivers. When receiving a signal, this sensor node uses the time difference of arrival at different antennas or ultrasound receivers to calculate the AoA of the signal.

RSSI [6] is defined as the amount of power in a received signal. The strength of the received signal decreases with the increase of propagation distance. Given that the signal power at the sender is known, the receiving sensor node can calculate the distance to the sender with the strength of the received signal.

ToA requires precise time synchronization between sensor nodes. TDoA, though less dependent on time synchronization, still needs precise timers. The computational complexity of calculating angles in AoA is large. Noise and fading are a major source of error in RSSI [8].

1.2.2 Range-free localization algorithms

There are several range-free localization algorithms, such as Centroid [9], distance vector-hop (DV-Hop) [10], Amorphous [11] and approximate point-in triangulation test (APIT) [12]. In Centroid [9], each unknown node calculates the coordinates as the centroid of the anchor nodes within the transmission radius (TR).

In DV-Hop [10], the minimum hop count (HC) between two sensor nodes is obtained via broadcast. An unknown node estimates the distance to an anchor node as the product of the minimum HC to this anchor node and the distance per hop (DPH)

value. Based on the estimated distances to the anchor nodes, each unknown node estimates the location.

Amorphous [11] is based on DV-Hop. It employs more complex ways of calculating the minimum HCs between the sensor nodes as well as estimating the DPH.

APIT [12] uses different sets of anchor nodes to segment the area of the WSN into a large number of triangles. An unknown node identifies the possible triangles that enclose its location via messages from the anchor nodes, and obtains the overlapping area of these triangles. This unknown node estimates the location as the center of gravity of this overlapping area.

The localization accuracy of APIT and Centroid depends largely on the AND of the WSN and the TR of the sensor nodes. The Amorphous algorithm requires an estimation of the average sensor node density of the WSN. DV-Hop is a distributed algorithm, and is easy to implement. However, the localization accuracy of DV-Hop needs to be improved [12].

1.3 Project objectives and contributions

Range-free algorithms, though less accurate than range-based schemes, have smaller computational complexity. Since range-free techniques do not use signal strength, they are more robust to noise and fading. Hence, range-free approaches are more robust. Among the range-free algorithms, DV-Hop is easy to implement but requires improvement in accuracy. Therefore, the factors affecting the performance of the DV-Hop algorithms should be studied.

This project focuses on the performance of DV-Hop and two variations of DV-Hop: DV-Hop with correction (CDV-Hop) [13] and improved DV-Hop (IDV-Hop) [14]. The evaluation of these algorithms considers multiple factors: the anchor node density (AND), the number of sensor nodes (SNN), the TR of the sensor nodes and the neighborhood size (NS) of the sensor nodes. The connected k-neighborhood (CKN) algorithm is used as a tool for the investigation of the NS on the performance of the three DV-Hop algorithms.

This report is organized as follows:

Chapter 1 provides a brief introduction to range-based and range-free localization algorithms, followed by an overview of the objectives and contributions of this project.

Chapter 2 gives the WSN mathematical model, and describes the DV-Hop, CDV-Hop, IDV-Hop and CKN algorithms.

Chapter 3 presents the implementation of the three DV-Hop algorithms using a WSN simulator called NetTopo.

Chapter 4 describes the experiments, compares the three DV-Hop algorithms, and discusses the factors affecting the performance of these algorithms.

Chapter 5 contains a conclusion of this project. It also presents the potential future work to improve and extend the current work.

Chapter 2

Algorithms

In this chapter, the algorithms used in this project are described. In Section 2.1 the WSN model is defined. Section 2.2 to Section 2.4 describe the three DV-Hop algorithms which are evaluated in this project. Section 2.5 describes the CKN algorithm.

2.1 WSN model

The connectivity of a WSN W is described by a graph $G = (S, E)$, where S is the set of sensor nodes in two-dimensional Euclidean space, and E describes the adjacency between sensor nodes.

$S = A \cup U$ and $A \cap U = \emptyset$, where A is the set of anchor nodes, and U is the set of unknown nodes. $M = |S|$, $N = |A|$ and $L = |U|$, so that $M = N + L$. If all sensor nodes in S have the same TR t, any two sensor nodes are neighbors if and only if the Euclidean distance between them is at most t. That is, for $\delta, \varepsilon \in S$, $\{\delta, \varepsilon\} \in E \Leftrightarrow d(\delta, \varepsilon) \leq t$, where $d(\delta, \varepsilon)$ is the Euclidean distance between δ and ε.

2.2 DV-Hop

DV-Hop is a distributed algorithm which is divided into three stages:

1. the broadcast of HCs from the anchor nodes;
2. the calculation and the broadcast of DPHs by the anchor nodes;
3. the estimation of the coordinates of the unknown nodes.

2.2.1 Stage 1 of DV-Hop

In Stage 1, each anchor node in A broadcasts an HC message to all sensor nodes in S. The message contains the coordinates of this anchor node and the HC to this anchor node. The HC is initialized to 0. Each sensor node maintains a hop table (HT). Each entry in the HT corresponds to an anchor node. A new entry is created when an HC message regarding a new anchor node is received. The entry contains the identification and the coordinates of this anchor node, and the HC to this anchor node.

Each time a sensor node $\varepsilon \in S$ receives an HC message regarding an anchor node $\alpha \in A$, ε updates the entry for α or creates an entry for α. If the HC in the message is less than the HC in the entry for α or α is a new anchor node, ε updates the entry for α with the HC from the message, and broadcasts the message with the HC increased by 1; otherwise, ε discards the message. The process ends when ε does not receive any HC messages after a certain period of time. At the end of this process, ε stores the minimum HCs to all anchor nodes that it can reach [10].

2.2.2 Stage 2 of DV-Hop

In Stage 2, each anchor node $\alpha \in A$ calculates the DPH dph_α as [10]

$$dph_\alpha = \frac{\sum_{j=1}^{j \le N_\alpha} d(\alpha, \beta_j)}{\sum_{j=1}^{j \le N_\alpha} h_j} \tag{2.1}$$

where $A_\alpha \subseteq A$ is the set of anchor nodes in the HT of α, $N_\alpha = |A_\alpha|$, $\beta_j \in A_\alpha$ for $1 \le j \le N_\alpha$, and h_j is the minimum HC from α to β_j.

Each anchor node broadcasts its DPH in a message to all unknown nodes in U. Each time an unknown node $\mu \in U$ receives a DPH message, if μ has not received a DPH message from any other sensor node, μ stores the DPH, and broadcasts the message; otherwise, μ discards the message. This process ends when μ does not receive any DPH messages after a certain amount of time. At the end of this process, μ stores a DPH if it can reach at least one anchor node [10].

2.2.3 Stage 3 of DV-Hop

For each unknown node $\mu \in U$, the stored DPH is denoted as dph_μ, $A_\mu \subseteq A$ is the set of anchor nodes in the HT of μ, $N_\mu = |A_\mu|$, and $\upsilon_i \in A_\mu$ for $1 \leq i \leq N_\mu$. μ estimates the distance λ_i to υ_i by [10]

$$\lambda_i = h_i \times dph_\mu \tag{2.2}$$

where h_i is the minimum HC from μ to υ_i.

To estimate (x, y), the coordinates of μ, suppose the coordinates of υ_i are (x_i, y_i). Let [10]

$$\lambda_i = d(\mu, \upsilon_i) = \sqrt{(x - x_i)^2 + (y - y_i)^2} \tag{2.3}$$

and

$$\mathbf{B} = -2 \times \begin{bmatrix} x_1 - x_{N_\mu} & y_1 - y_{N_\mu} \\ x_2 - x_{N_\mu} & y_2 - y_{N_\mu} \\ \vdots & \vdots \\ x_{N_\mu - 1} - x_{N_\mu} & y_{N_\mu - 1} - y_{N_\mu} \end{bmatrix} \tag{2.4}$$

$$\mathbf{p} = \begin{bmatrix} \lambda_1^2 - \lambda_{N_\mu}^2 - x_1^2 + x_{N_\mu}^2 - y_1^2 + y_{N_\mu}^2 \\ \lambda_2^2 - \lambda_{N_\mu}^2 - x_2^2 + x_{N_\mu}^2 - y_2^2 + y_{N_\mu}^2 \\ \vdots \\ \lambda_{N_\mu - 1}^2 - \lambda_{N_\mu}^2 - x_{N_\mu - 1}^2 + x_{N_\mu}^2 - y_{N_\mu - 1}^2 + y_{N_\mu}^2 \end{bmatrix} \tag{2.5}$$

$$\mathbf{v} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.6}$$

Using the method of least squares (LS), by minimizing $\|\mathbf{Bv} - \mathbf{p}\|$, $\mathbf{v}$ is estimated as [10]

$$\mathbf{v} = (\mathbf{B^T B})^{-1} \mathbf{B^T p} \tag{2.7}$$

2.3 CDV-Hop

CDV-Hop also has three stages. Stages 1 and 2 of CDV-Hop are the same as Stages 1 and 2 of DV-Hop, whereas Stage 3 of CDV-Hop is different from Stage 3 of DV-Hop. Typically the paths between sensor nodes are not straight lines, so the original DV-Hop method introduces some error in (2.2). To improve the accuracy of the distance

estimation, CDV-Hop adds a correction so that

$$\lambda_i = h_i \times dph_\mu + c_i = h_i \times dph_\mu + \frac{h_i \times (t - dph_\mu)}{t} \tag{2.8}$$

where c_i is the correction factor, and t is the TR [13].

2.4 IDV-Hop

IDV-Hop also has three stages. Stage 1 of IDV-Hop is the same as Stage 1 of DV-Hop, whereas Stages 2 and 3 of IDV-Hop differ from Stages 2 and 3 of DV-Hop, and are described below.

2.4.1 Stage 2 of IDV-Hop

After the calculation of the DPH by (2.1), each anchor node broadcasts the DPH value in a message to all unknown nodes in U. Each unknown node $\mu \in U$ stores the DPHs from different anchor nodes. This process ends when μ does not receive any DPH messages after a certain period of time. At the end of this process, μ stores the DPHs from all anchor nodes in A_μ.

2.4.2 Stage 3 of IDV-Hop

Each unknown node $\mu \in U$ calculates the average DPH dph_μ as [14]

$$dph_\mu = \frac{\sum_{i=1}^{i \leq N_\mu} dph_{\upsilon_i}}{N_\mu} \tag{2.9}$$

where dph_{υ_i} is the DPH from υ_i. μ estimates the distance λ_i to υ_i using (2.2). From (2.3), $x_i^2 + y_i^2 - 2x_i x - 2y_i y + x^2 + y^2 = \lambda_i^2$, so that [14]

$$\lambda_i^2 - r_i = -2x_i x - 2y_i y + r \tag{2.10}$$

where $r_i = x_i^2 + y_i^2$, and $r = x^2 + y^2$.

Let

$$\mathbf{z} = \begin{bmatrix} x \\ y \\ r \end{bmatrix} \tag{2.11}$$

$$\mathbf{P} = \begin{bmatrix} -2x_1 & -2y_1 & 1 \\ -2x_2 & -2y_2 & 1 \\ \vdots & \vdots & \vdots \\ -2x_{N_\mu} & -2y_{N_\mu} & 1 \end{bmatrix} \tag{2.12}$$

$$\mathbf{q} = \begin{bmatrix} \lambda_1^2 - r_1 \\ \lambda_2^2 - r_2 \\ \vdots \\ \lambda_{N_\mu}^2 - r_{N_\mu} \end{bmatrix} \tag{2.13}$$

Using the method of LS, by minimizing $\|\mathbf{Pz} - \mathbf{q}\|$, $\mathbf{z}$ is estimated as [14]

$$\mathbf{z} = (\mathbf{P^T P})^{-1}\mathbf{P^T q} \tag{2.14}$$

2.5 CKN

In CKN, an awake sensor node means that this sensor node participates in the WSN, whereas a sleeping sensor node does not participate in the WSN. Given a value of k, CKN selects a subset of awake sensor nodes $C \subseteq S$. The selected subset of awake sensor nodes C satisfies the following conditions:

1. Each sensor node $\sigma \in S$ has at least $min(k, De_\sigma)$ awake neighbors from C, where De_σ is the number of neighbors of σ from S.

2. The sensor nodes in C are connected if the sensor nodes in S are connected.

3. No subset of S satisfying the above two conditions has a smaller number of sensor nodes than C.

The CKN algorithm provides a near-optimal solution to selecting the sensor nodes satisfying the above conditions[15]. The NS of the selected sensor nodes approximates k. In CKN, time is divided into discrete epochs, and the subsets of awake sensor nodes selected by CKN differ on epochs. CKN is a distributed algorithm. On each epoch, CKN is depicted as follows [15].

Algorithm 1 CKN

For each sensor node $\sigma \in S$

1: Pick a uniformly distributed random rank $rank_\sigma$ between 0 and 1 for σ.

2: Broadcast $rank_\sigma$ and receive the ranks of its currently awake neighbors Ne_σ. Let R_σ be the set of these ranks.

3: Broadcast R_σ and receive R_ρ from each $\rho \in Ne_\sigma$.

4: If $|Ne_\sigma| < k$ or $|Ne_\rho| < k$ for any $\rho \in Ne_\sigma$, σ remains awake. Return.

5: Compute $C_\sigma = \{\rho | \rho \in Ne_\sigma \wedge rank_\rho < rank_\sigma\}$.

6: σ goes to sleep if both the following conditions satisfies; otherwise, σ remains awake.

- Any two sensor nodes in C_σ are connected either directly or through sensor nodes satisfying both the following conditions:
  - within 2-hop neighborhood of σ;
  - with ranks less than $rank_\sigma$.
- Any sensor node in Ne_σ has at least k neighbors from C_σ.

7: Return.

Chapter 3

Implementation

This chapter presents the implementation of the three DV-Hop algorithms. Section 3.1 gives an overview of the WSN simulators. Section 3.2 introduces NetTopo. Section 3.3 discusses the implementation of the three DV-Hop algorithms, including the representation of sensor nodes in NetTopo, the integration of the three algorithms into NetTopo and the principle of code reusability in the implementation.

3.1 WSN simulators

With the recent development in WSNs, a large number of WSN simulators have been developed. These simulators can be classified into the following three categories:

1. algorithm-level;
2. packet-level;
3. instruction-level.

Algorithm-level simulators focus on the logic and the representation of the algorithms. AlgoSenSim [16] is a framework for distributed algorithms in WSNs. Sinalgo [17] captures the view of the actual network devices by offering a message passing view of the network. The main goal of Shawn [18] is to create an abstract, repeatable and expressive approach to the simulation. NetTopo [19] provides an extensible integrated framework of simulation and visualization, and can integrate with real WSN testbeds.

Packet-level simulators implement different layers of the network stack. NS-2 [20] is a general networking simulator which supports WSN simulation. SensorSim [21]

is a framework based on NS-2. SensorSim provides a battery model, a radio model, a CPU model and a sensor device model. J-Sim [22] is derived from SensorSim, and supports real-time process-driven simulation. GloMoSim [23] is a parallel discrete-event simulator.

Instruction-level simulators simulate the execution of the CPU at the level of instructions or clock cycles, and can be regarded as emulators. TOSSIM [24] is compiled directly from TinyOS code, and simulates the execution of TinyOS. Atemu [25] provides the emulation on AVR processor based systems. Avrova [26] is a collection of analysis tools for programs on the AVR microcontrollers made by Atmel and the Mica2 sensor nodes.

3.2 NetTopo simulator

NetTopo is an open source WSN simulator focusing on the visualization of the connection and sensing data of sensor nodes, and the topology of the WSN. In NetTopo, users can define and extend the processing behavior of sensor nodes. The extensibility of NetTopo benefits the simulation of various localization algorithms for WSNs. Therefore, this project uses NetTopo for the simulation of the three DV-Hop algorithms.

The architecture of NetTopo consists of several components, as shown in Figure 3.1. The *Virtual WSN* component is a representation of the WSN in the simulator. NetTopo models sensor nodes in the *Node* component. The *Algorithm* component includes algorithms in WSNs. The algorithms range from routing, clustering, scheduling and controlling algorithms to localization algorithms [19].

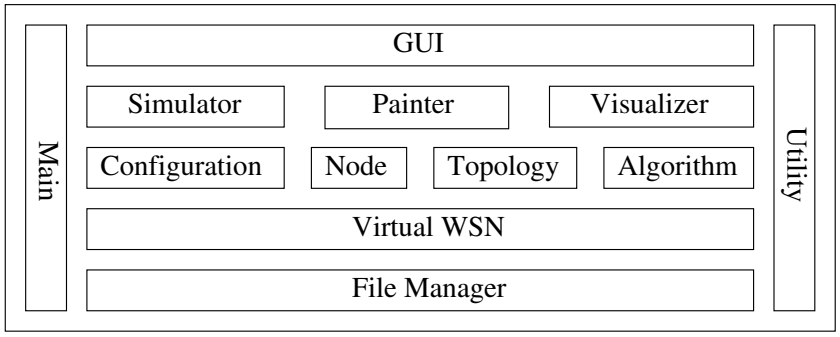

Figure 3.1: NetTopo Architecture.

Figure 3.2 shows that the *graphical user interface* (*GUI*) component of NetTopo

has three parts, a graphical display canvas, a property tab and a console. The graphical display canvas shows the deployment of the sensor nodes. Different types of sensor nodes are displayed in different colors. When the user clicks a sensor node on the canvas, the property tab displays the properties of this sensor node. Real-time logs are written to the console of the GUI at run-time [19].

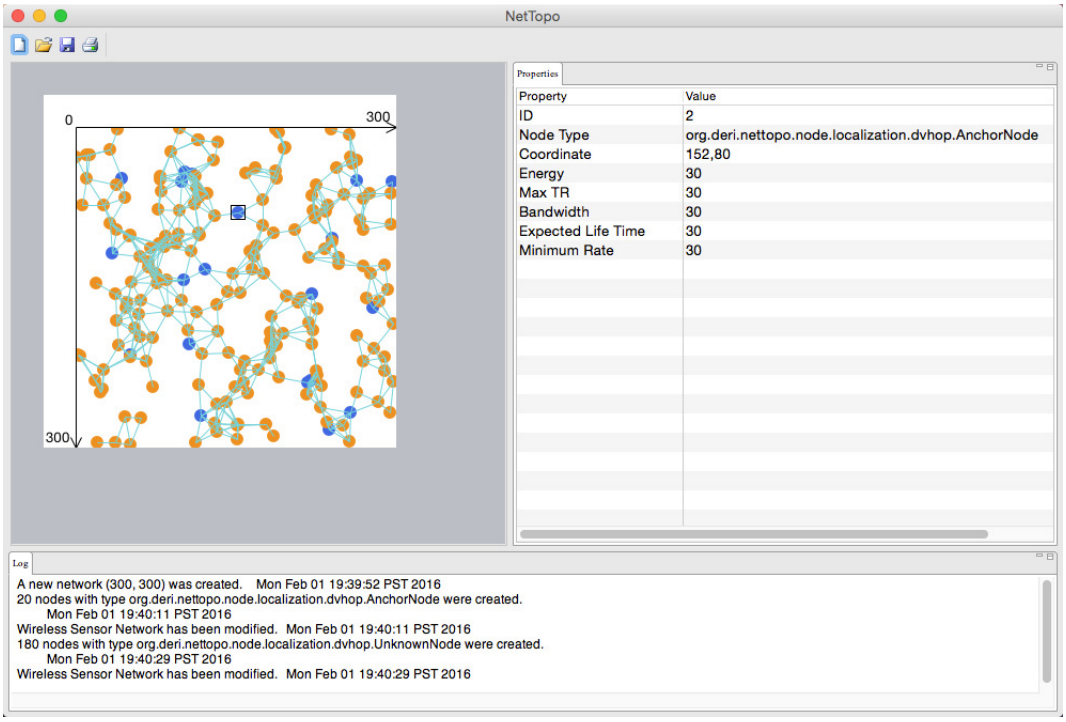

Figure 3.2: Graphical user interface of NetTopo.

Figure 3.3 shows an example of the interaction between a user and the components in NetTopo. NetTopo takes 18 steps to respond to a request from a user. On receiving a request in Step 1, NetTopo uses the *GUI* component to run the simulation from the *Main* component in Step 2 . The *Main* component interacts with the *Algorithm* component in Steps 3-5. The *Algorithm* component requests the sensor nodes from the *Virtual WSN* component in Step 6. The *Virtual WSN* component returns the sensor nodes to the *Algorithm* component in Step 7. Each sensor node in the *Node* component runs the algorithm in step 8, and produces the results. The results are forwarded from the *Node* component to the *Algorithm* component and the *Main* component in Steps 10 and 11. After receiving the results, the *Main* component interacts with the *Painter* component to refresh the canvas of the *GUI* component in Steps 12-17. The *GUI* component notifies the user of the results in Step 18 [19].

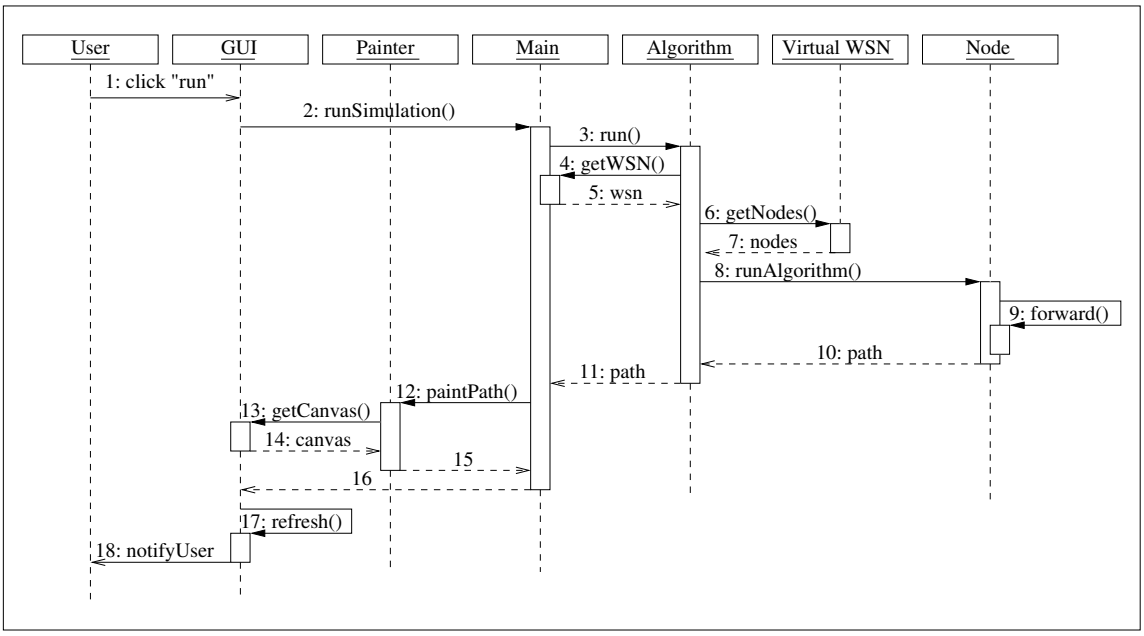

Figure 3.3: Sequence diagram of simulation in NetTopo.

3.3 Implementation of DV-Hop algorithms

Sensor nodes have different sensing data and functionalities in different applications, hence sensor nodes can have different representations. NetTopo declares an interface called `VNode`, which defines several methods modeling the common behavior of a sensor node. Users can customize their own classes of sensor nodes by implementing the `VNode` interface. The `WirelessSensorNetwork` class models a WSN containing zero or more sensor nodes.

Figure 3.4 shows that NetTopo implements two classes of sensor nodes, the `AnchorNode` and the `UnknownNode` classes for DV-Hop algorithms. The two classes are derived from the `DVHopNode` class. The `DVHopNode` class extends the `SensorNode` class, and implements the `VNode` interface. In addition to the functionalities of the `DVHopNode` class, the `AnchorNode` class calculates and broadcasts the DPHs. The `UnknownNode` class handles the DPH messages, and estimates the coordinates.

From Figure 3.5, a WSN algorithm is represented by a class implementing the `Algorithm` interface. Each algorithm has several functionalities. Each functionality of the algorithm is represented by a class implementing the `AlgorFunc` interface.

The three DV-Hop algorithms are implemented in the `Algor_DVHOP`, the `Algor_C-DVHOP` and the `Algor_IDVHOP` classes respectively. The functionalities of CDV-Hop

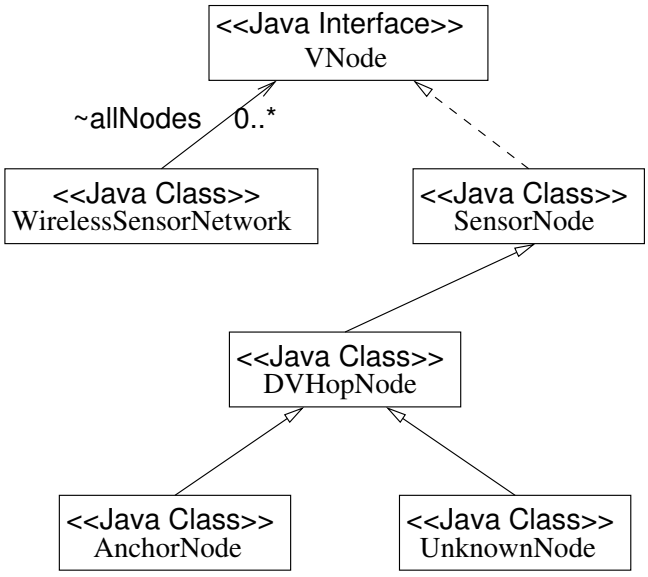

Figure 3.4: Class diagram of sensor nodes in NetTopo.

and IDV-Hop are implemented in the `CDVHOP_Run_CDVHop` and the `IDVHOP_Run_IDVHop` classes respectively. Since both CDV-Hop and IDV-Hop share some features of DV-Hop, the `CDVHOP_Run_CDVHop`, and the `IDVHOP_Run_IDVHop` classes are derived from the `DVHOP_Run_DVHop` class.

As stated in Section 2, all three DV-Hop algorithms have a routine of three stages. CDV-Hop and IDV-Hop reimplement some of the functions in the three stages. As shown in Figure 3.6, some initialization work is done in the first four functions before the three stages.

The `DVHOP_Run_DVHop` class implements the broadcast of Stage 1 in the `BroadcastHCs()` function (Section 2.2.1). The calculation of DPH in Stage 2 is implemented in the `CalculateDPH()` function and the broadcast of the DPHs in Stage 2 is implemented in the `BroadcastDPHs()` function (Section 2.2.2). The estimation of the coordinates by the unknown nodes in Stage 3 is implemented in the `EstimateCoordinates()` function (Section 2.2.3).

The `CDVHOP_Run_CDVHop` class overrides the `DeduceCoordinates()` function by adding a correction to the estimated distance between an unknown node and an anchor node before the estimation of the coordinates (Section 2.3).

The `IDVHOP_Run_IDVHop` class overrides the `BroadcastDPHs()` and the `EstimateCoordinates()` functions. In the `BroadcastDPHs()` function, each unknown node stores the DPHs from all anchor nodes in the HT (Section 2.4.1); the `EstimateCoordi-`

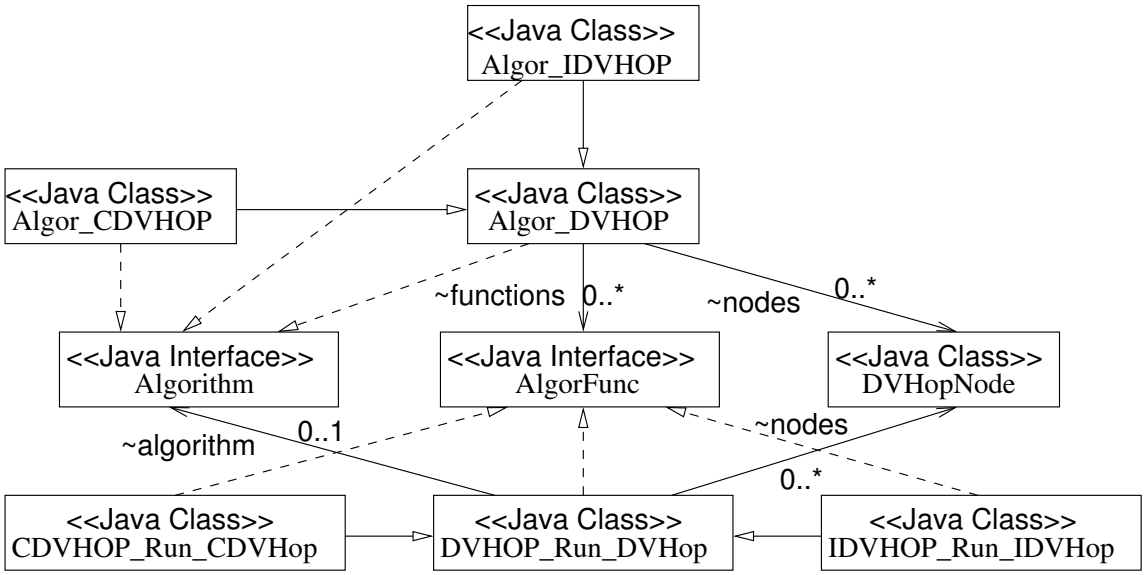

Figure 3.5: Class diagram of DV-Hop algorithms.

`nates()` function reimplements the estimation of the coordinates by the unknown nodes (Section 2.4.2).

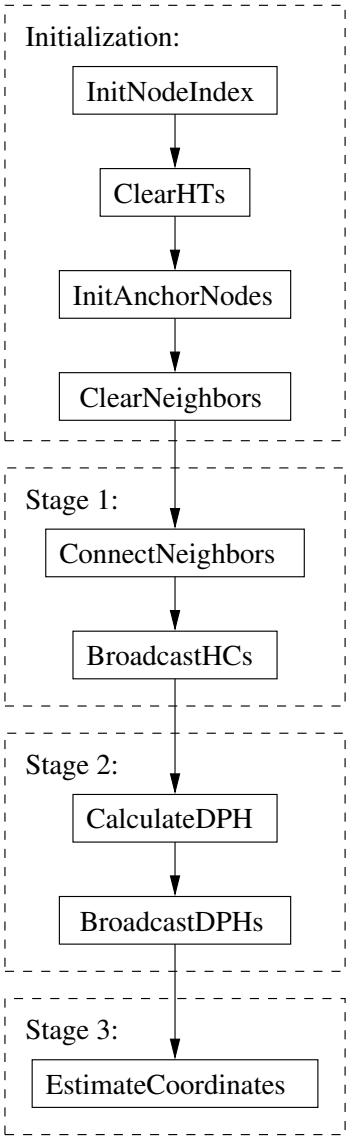

Figure 3.6: Flow chart of DV-Hop algorithms.

Chapter 4

Experiments and Discussion

This chapter presents the experiments and the discussion on the three algorithms. Section 4.1 investigates the computational complexity of the three algorithms. Section 4.2 defines the localization error of a set of unknown nodes. The localization error is used to evaluate the accuracy of the DV-Hop algorithms. The following three experiments are conducted in this project:

1. Experiment 1 investigates the impact of the AND and the TR on the accuracy of the three algorithms.

2. Experiment 2 investigates the impact of the SNN and the TR on the accuracy of the three algorithms.

3. Experiment 3 investigates the impact of the AND and the NS on the accuracy of the three algorithms.

4.1 Computational complexity

The three DV-Hop algorithms are distributed, hence the computational complexity of the algorithms is considered for each sensor node. The anchor nodes and the unknown nodes behave differently, hence the computational complexity is considered separately. For the anchor nodes, the computational complexity of the DPH calculation is considered; for the unknown nodes, the computational complexity of the estimation of the coordinates is considered. For an anchor node $\alpha \in A$, $|A_\alpha| \leq N$, so the computational complexity of the DPH calculation by the three algorithms is upper bounded by $\mathcal{O}(N)$ for the anchor nodes, according to (2.1). For an unknown

node $\mu \in U$, $|A_\mu| \leq N$, so the computational complexity of the estimation of the coordinates by the three algorithms is upper bounded by $\mathcal{O}(N^3)$, according to (2.2), (2.7), (2.8), (2.9) and (2.14).

4.2 Localization error

To quantify the accuracy of a localization algorithm, the localization error of an unknown node τ_i is defined as

$$error_i = \sqrt{\Delta x_i{}^2 + \Delta y_i{}^2} \tag{4.1}$$

where $\tau_i \in U$, for $1 \leq i \leq L$, Δx_i and Δy_i are the errors in the estimated coordinates of τ_i.

The localization error of U is defined as the average localization error of all unknown nodes in U

$$\overline{error} = \frac{\sum_{i=1}^{L} error_i}{L} \tag{4.2}$$

4.3 Anchor node density and transmission radius

The procedure of the experiment on the impact of the AND and the TR is described in Algorithm 2. The size of the WSNs in this experiment is 500 m$\times$ 500 m. The SNN of all WSNs is 400. The ANDs of the WSNs are 3%, 4%, 5%, 6%, 7%, 8%, 9%, 10%, 20% and 30%. The TRs of all sensor nodes in the WSNs are 45 m, 60 m, 75 m, 90 m, 105 m, 120 m, 135 m and 150 m. In two nested for-loops, the outer for-loop iterates all ANDs, the inner for-loop iterates all TRs. DV-Hop, CDV-Hop and IDV-Hop are run on 200 WSNs, and calculate the average localization error of the 200 WSNs, for each pair of AND and TR.

Algorithm 2 Procedure of the experiment on the impact of the TR and the AND

```
The size of the WSNs is set to 500 m× 500 m.
The SNN of all WSNs is set to 400.
for AND in {3%, 4%, 5%, 6%, 7%, 8%, 9%, 10%, 20%, 30%} do
    for TR in {45 m, 60 m, 75 m, 90 m, 105 m, 120 m, 135 m, 150 m} do
        for W_i in {W_1, W_2, ..., W_200} do
            Run DV-Hop on W_i.
            Run CDV-Hop on W_i.
            Run IDV-Hop on W_i.
        end for
        DV-Hop calculates the average localization error of the 200 WSNs.
        CDV-Hop calculates the average localization error of the 200 WSNs.
        IDV-Hop calculates the average localization error of the 200 WSNs.
    end for
end for
```

Figures 4.1-4.3 show the localization error of DV-Hop, CDV-Hop and IDV-Hop respectively, when the AND and the TR vary. For all three algorithms, given a TR, an increase in the AND decreases the localization error in most cases. The exceptions are listed in Table 4.1. Given an AND, when the TR is smaller than 60 m, an increase

Algorithm	**TR**	**Increase in AND**	**Increase in localization error**
DV-Hop	45 m	20% to 30%	32.9 m to 33.2 m
DV-Hop	75 m	9% to 10%	27.3 m to 27.4 m
CDV-Hop	45 m	20% to 30%	33.3 m to 33.6 m

Table 4.1: Cases of increase in localization error with increase in anchor node density.

in the TR decreases the localization error of the three algorithms; when the TR is larger than 60 m, an increase in the TR decreases the localization error. Therefore, 60 m is the best TR in this experiment.

When the TR is smaller than this best value, an increase in the TR improves the connectivity of the WSN. Hence, each unknown node can hear from more anchor nodes, and stores more entries in the HT. However, after the WSN is fully connected, an increase in the TR decreases the minimum HCs between sensor nodes, increases the DPH calculated in (2.1), and increases the quantization error in (2.2).

Figures 4.4-4.11 compare the localization error of the three algorithms when the

AND varies. Compared with DV-Hop, IDV-Hop consistently shows lower localization error. IDV-Hop has an average of 10% less localization error than DV-Hop. However, CDV-Hop shows overall larger localization error than DV-Hop when the TR is in the range of 90 m and 150 m, and shows smaller localization error than DV-Hop when the TR is 60 m. The curves of DV-Hop and CDV-Hop overlap when the TR is 45 m or 75 m.

4.4 Number of sensor nodes and transmission radius

The procedure of the experiment on the impact of the SNN and the TR is described in Algorithm 3. The size of the WSNs in this experiment is 500 m × 500 m. The AND of the WSNs is 10%. The SNNs of the WSNs are 300, 400, 500, 600, 700 and 800. The TRs of all sensor nodes in the WSNs are 30 m, 45 m, 60 m, 75 m, 90 m, 105 m, 120 m, 135 m and 150 m. In two nested for-loops, the outer for-loop iterates all SNNs, the inner for-loop iterates all TRs. DV-Hop, CDV-Hop and IDV-Hop are run on 200 WSNs, and calculate the average localization error of the 200 WSNs, for each pair of SNN and TR.

Algorithm 3 Procedure of the experiment on the impact of the SNN and the TR

```
1: The size of the WSNs is set to 500 m× 500 m.
2: The ANDs of the WSNs is set to 10%.
3: for SNN in {300, 400, 500, 600, 700, 800} do
4:     for TR in {30 m, 45 m, 60 m, 75 m, 90 m, 105 m, 120 m, 135 m, 150 m} do
5:         for W_i in {W_1, W_2, ..., W_200} do
6:             Run DV-Hop on W_i.
7:             Run CDV-Hop on W_i.
8:             Run IDV-Hop on W_i.
9:         end for
10:        DV-Hop calculates the average localization error of the 200 WSNs.
11:        CDV-Hop calculates the average localization error of the 200 WSNs.
12:        IDV-Hop calculates the average localization error of the 200 WSNs.
13:    end for
14: end for
```

Figures 4.12-4.14 show the localization error of DV-Hop, CDV-Hop and IDV-Hop respectively, when the SNN and the TR vary. The best TRs for the three algorithms are 75 m, 60 m, 60 m, 45 m, 45 m and 45 m when the SNNs are 300, 400, 500, 600, 700 and 800 respectively. An increase in the SNN decreases the best TR.

4.5 Anchor node density and neighborhood size

CKN is used as a tool in the investigation of the NS on the performance of the three DV-Hop algorithms. On an epoch, CKN selects a subset of awake sensor nodes from the WSN. The network of the awake sensor nodes selected by CKN approximates a network with the NS being k [15]. With CKN being run with different values of k, networks with different values of NS are generated. Since CKN is a random algorithm, multiple epochs of CKN are run with the same value of k on each WSN in this experiment.

The procedure of the experiment on the impact of the AND and the NS is described in Algorithm 4. The size of the WSNs in this experiment is 500 m $\times$ 500 m. The ANDs are 5%, 6%, 7%, 8%, 9%, 10%, 20% and 30%. The NS's are 1, 3, 5, 7 and ∞. The first for-loop iterates all ANDs. The second for-loop creates 20 WSNs, for each AND. The third for-loop iterates all NS's, for each WSN. In the fourth for-loop, CKN is run for 50 epochs on each WSN, for each pair of AND and NS. DV-Hop, CDV-Hop and IDV-Hop are run on the network of the awake sensor nodes, on each epoch. DV-Hop, CDV-Hop and IDV-Hop calculate the average localization error of the 50 epochs of the 20 WSNs, for each pair of AND and NS.

Algorithm 4 Procedure of the experiment on the impact of the AND and the NS

1: The size of the WSNs is set as 500 m× 500 m.
2: **for** AND in {5%, 6%, 7%, 8%, 9%, 10%, 20%, 30%} **do**
3: **for** W_i in {W_1, W_2, ..., W_{20}} **do**
4: **for** NS in {1, 3, 5, 7, ∞} **do**
5: **for** epoch in {$epoch_1$, $epoch_2$, ..., $epoch_{50}$} **do**
6: Run CKN.
7: Run DV-Hop on W_i.
8: Run CDV-Hop on W_i.
9: Run IDV-Hop on W_i.
10: **end for**
11: **end for**
12: **end for**
13: DV-Hop calculates the average localization error of the 50 epochs of the 20 WSNs for each pair of AND and NS.
14: CDV-Hop calculates the average localization error of the 50 epochs of the 20 WSNs for each pair of AND and NS.
15: IDV-Hop calculates the average localization error of the 50 epochs of the 20 WSNs for each pair of AND and NS.
16: **end for**

Figures 4.15-4.17 show the localization error of DV-Hop, CDV-Hop and IDV-Hop respectively, when the SN and the TR vary. When the NS is ∞, the extracted set of sensor nodes is the set of all sensor nodes in the WSN. Hence, the NS being set as ∞ is equivalent to not running CKN. The average number of awake sensor nodes are 125, 265, 355, 391 and 400 when the NS's are 1, 3, 5, 7 and ∞ respectively.

Given an algorithm and an AND, an increase in the NS decreases the localization error when the NS is in the range of 1 and 5. The curves overlap when the NS's are 5, 7 and ∞. The average of the decrease in the localization error versus the algorithm and the increase in the NS are listed in Table 4.2. The table shows that when the NS increases from 1 to 3, the decrease in the localization error is around 40%. It is noticeable that when the NS increases from 3 to 5, the average number of awake sensor nodes increases by 34%, but the decrease in the localization error of the three algorithms is in the range of 6% and 10%. According to the computational complexity of the estimation of the coordinates in Section 4.1, the estimated running

times of all three algorithms increase significantly.

Increase in NS	1 to 3	3 to 5	5 to 7	7 to ∞
Algorithm				
DV-Hop	43%	9.6%	1.35%	0%
CDV-Hop	40%	6.6%	0.82%	0%
IDV-Hop	41%	6.6%	0.79%	0%

Table 4.2: Decrease in localization error versus algorithm and increase in neighborhood size.

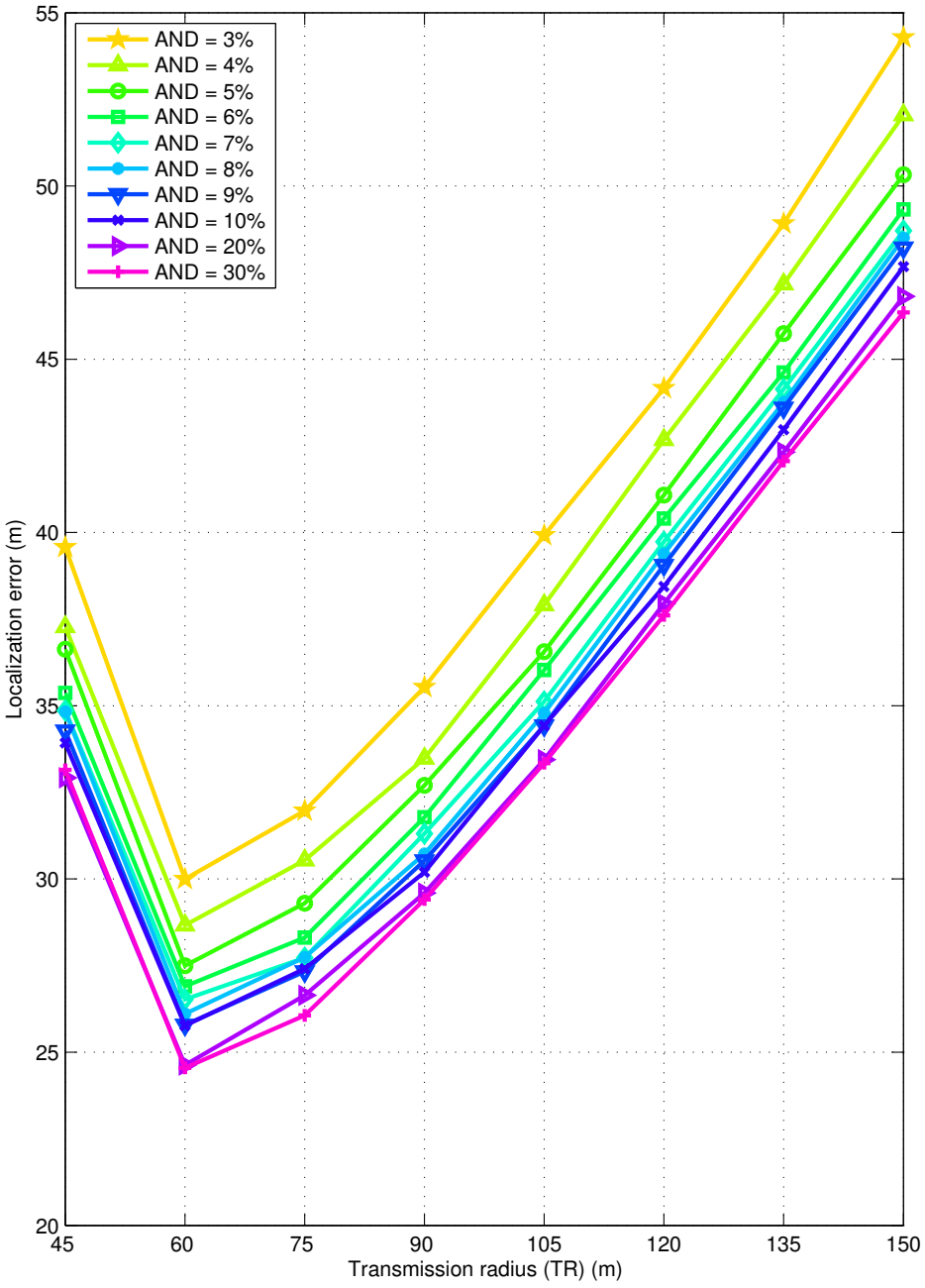

Figure 4.1: Localization error of DV-Hop versus transmission radius and anchor node density.

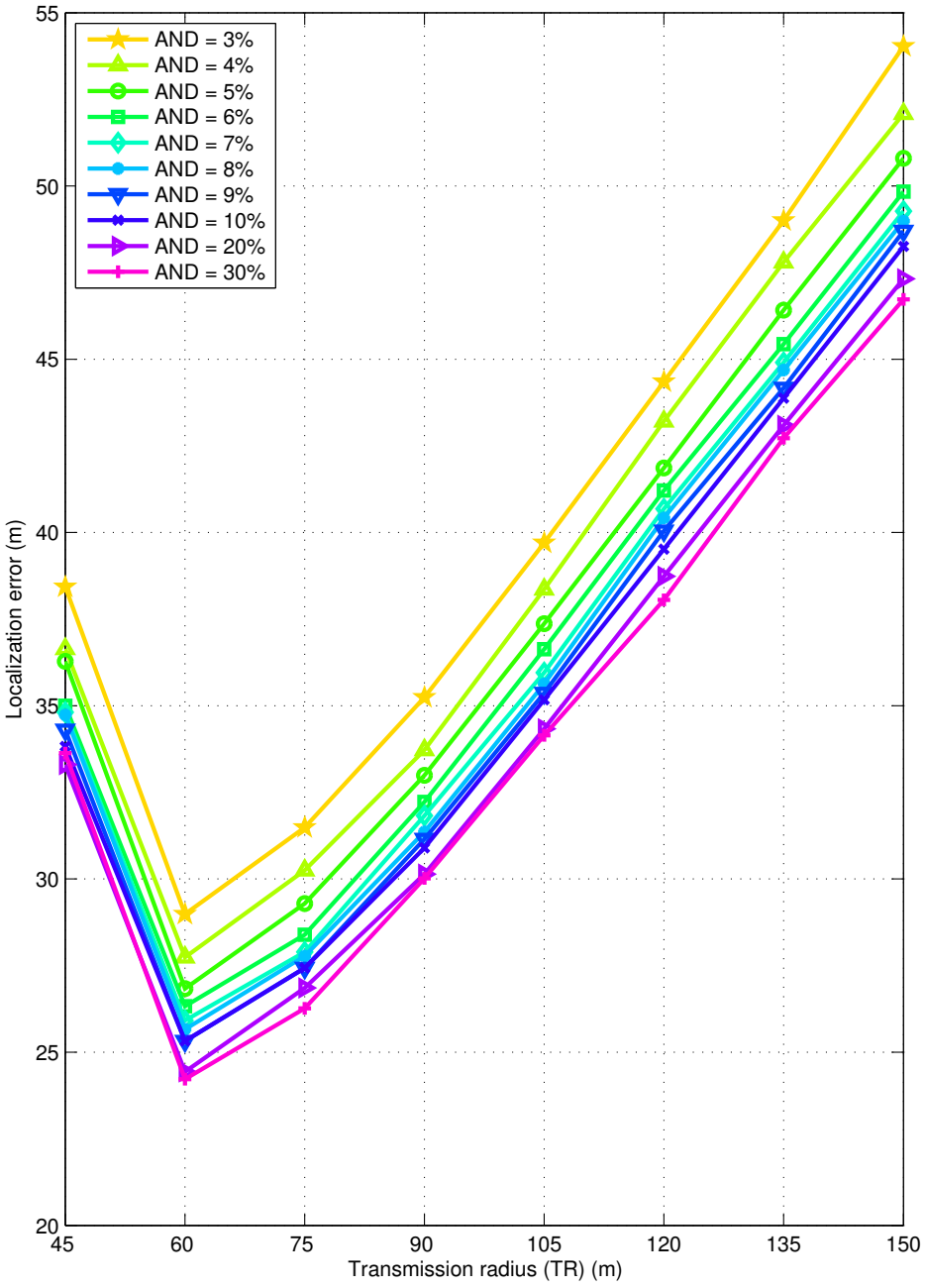

Figure 4.2: Localization error of CDV-Hop versus transmission radius and anchor node density.

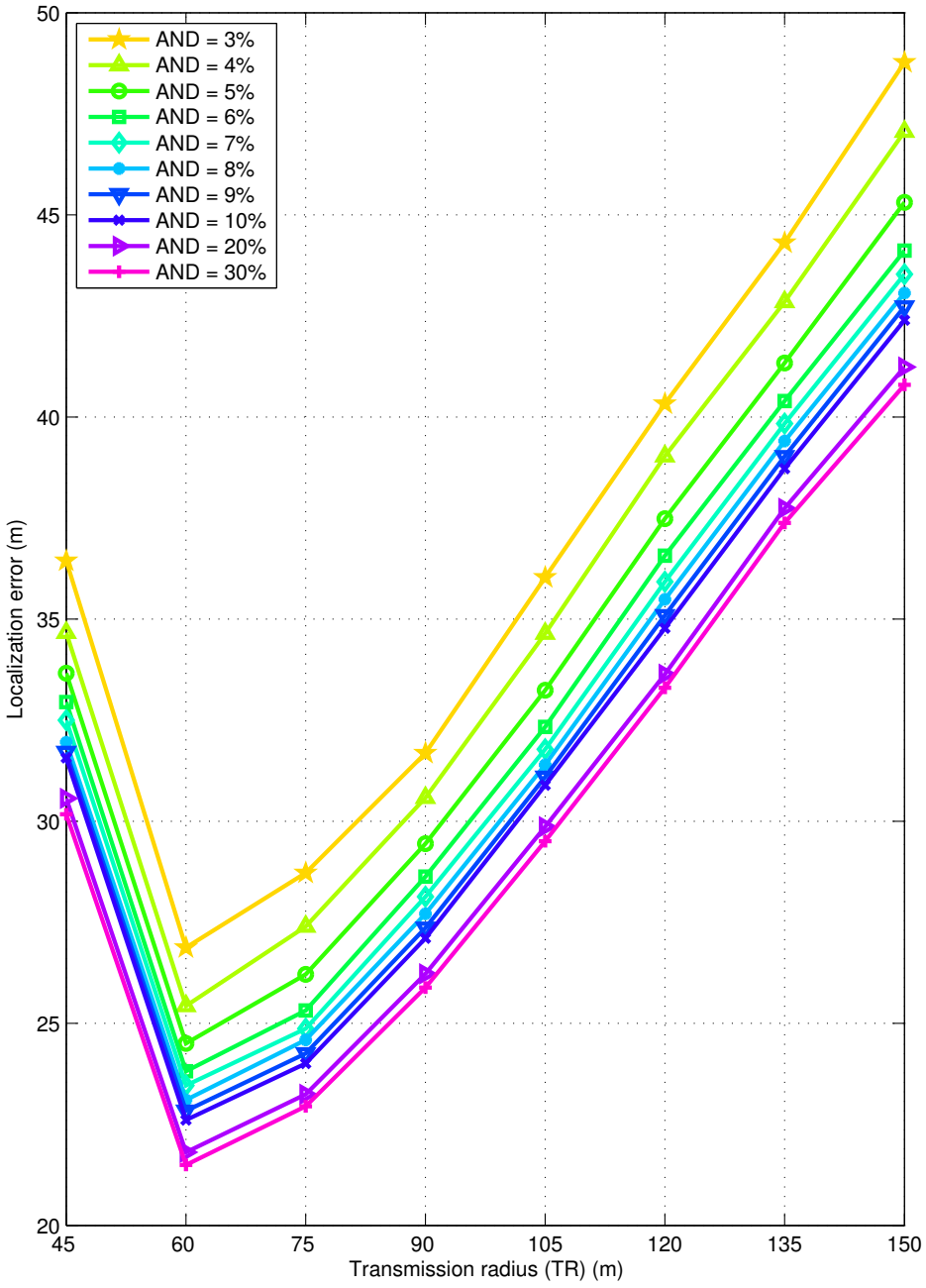

Figure 4.3: Localization error of IDV-Hop versus transmission radius and anchor node density.

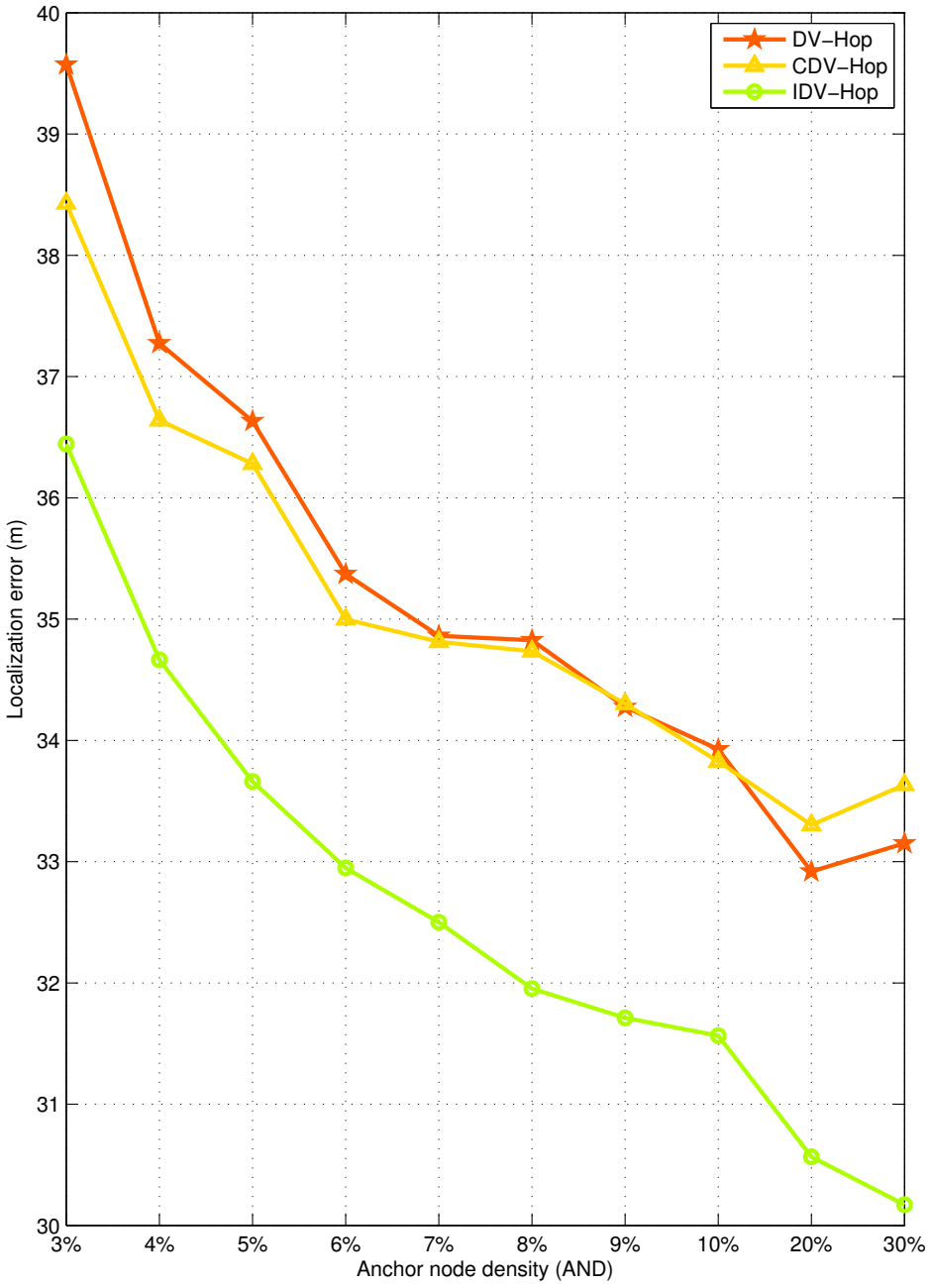

Figure 4.4: Comparison of localization error of DV-Hop, CDV-Hop and IDV-Hop when transmission radius is 45 m and anchor node density varies.

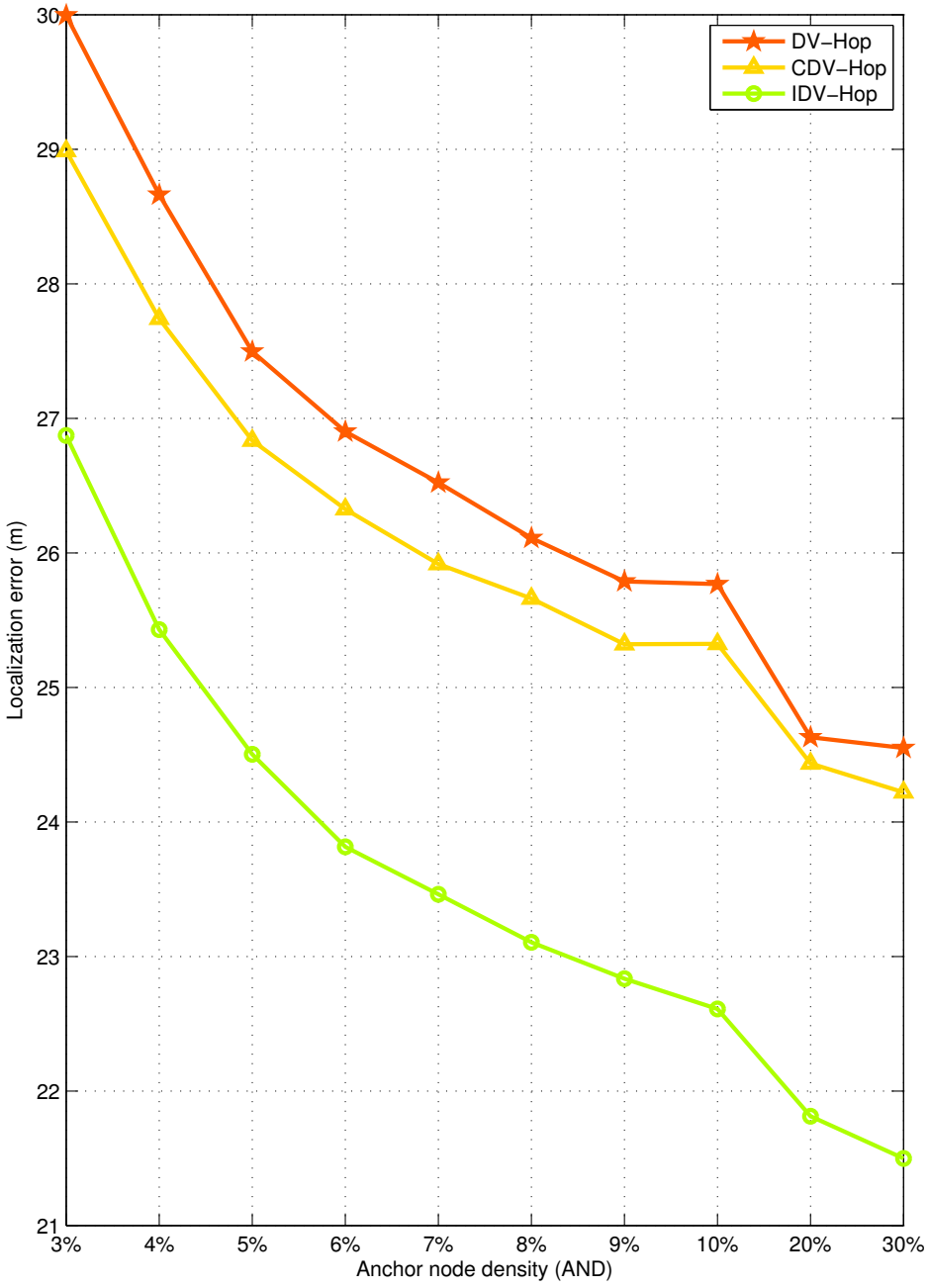

Figure 4.5: Comparison of localization error of DV-Hop, CDV-Hop and IDV-Hop when transmission radius is 60 m and anchor node density varies.

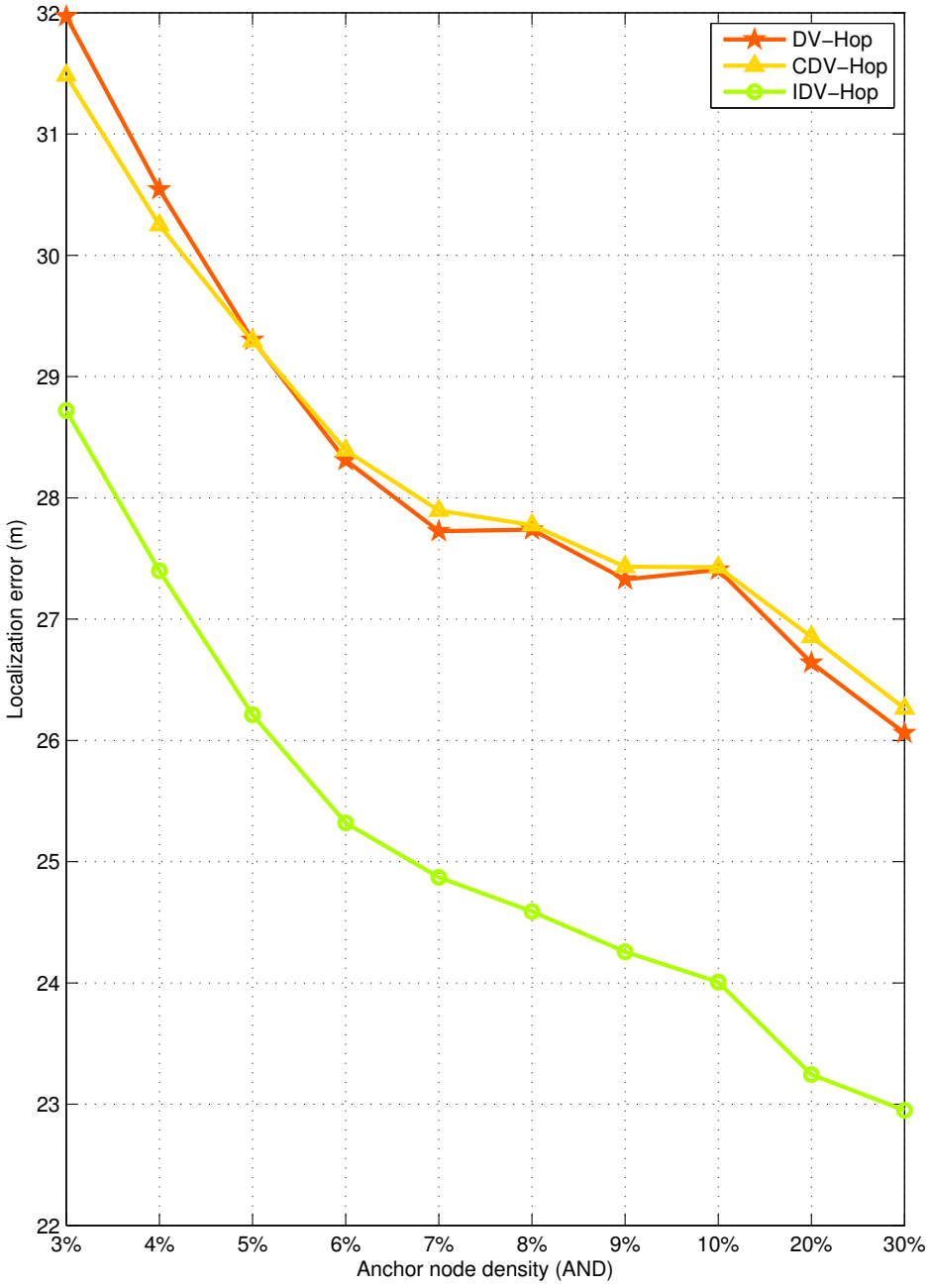

Figure 4.6: Comparison of localization error of DV-Hop, CDV-Hop and IDV-Hop when transmission radius is 75 m and anchor node density varies.

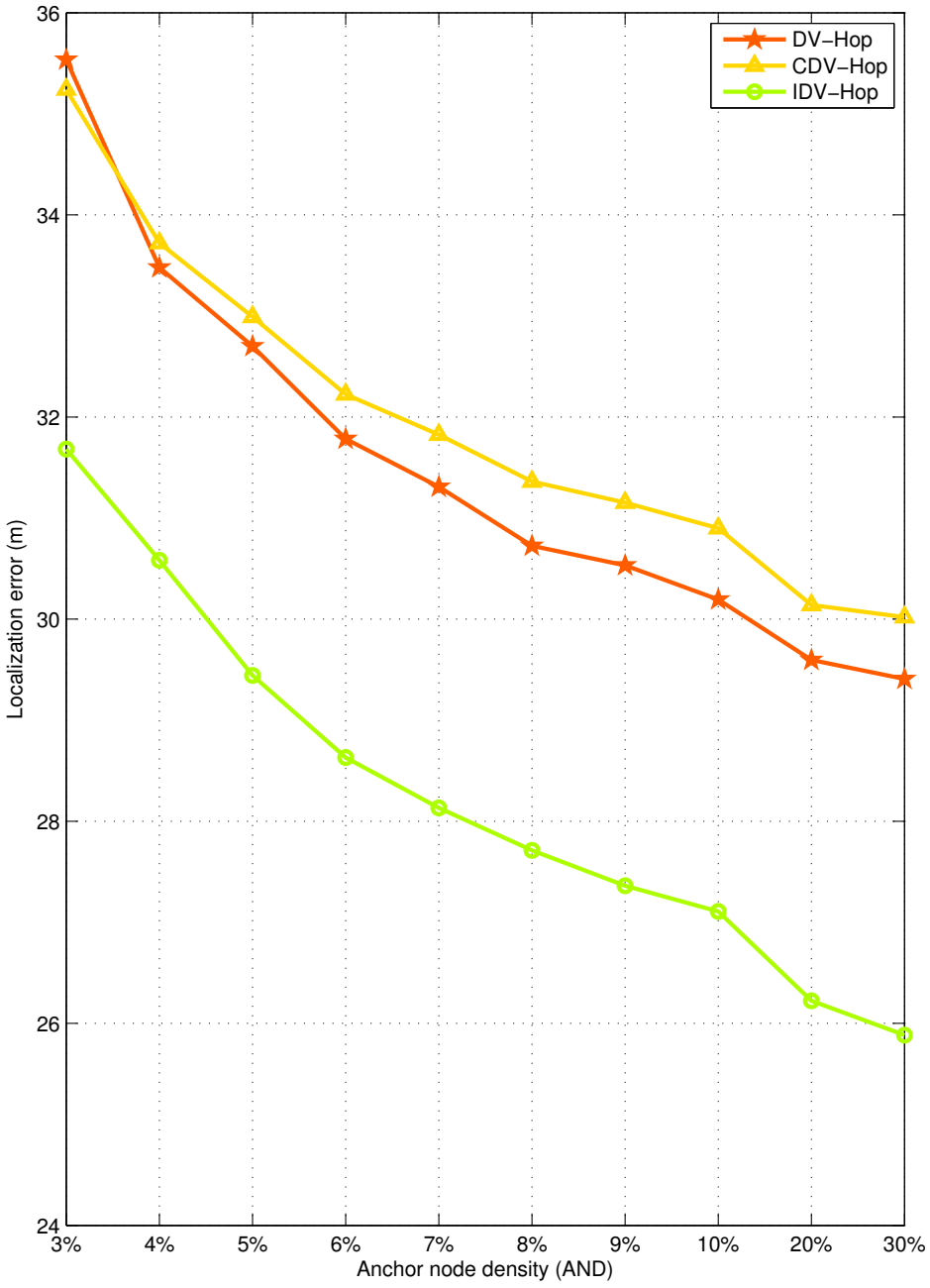

Figure 4.7: Comparison of localization error of DV-Hop, CDV-Hop and IDV-Hop when transmission radius is 90 m and anchor node density varies.

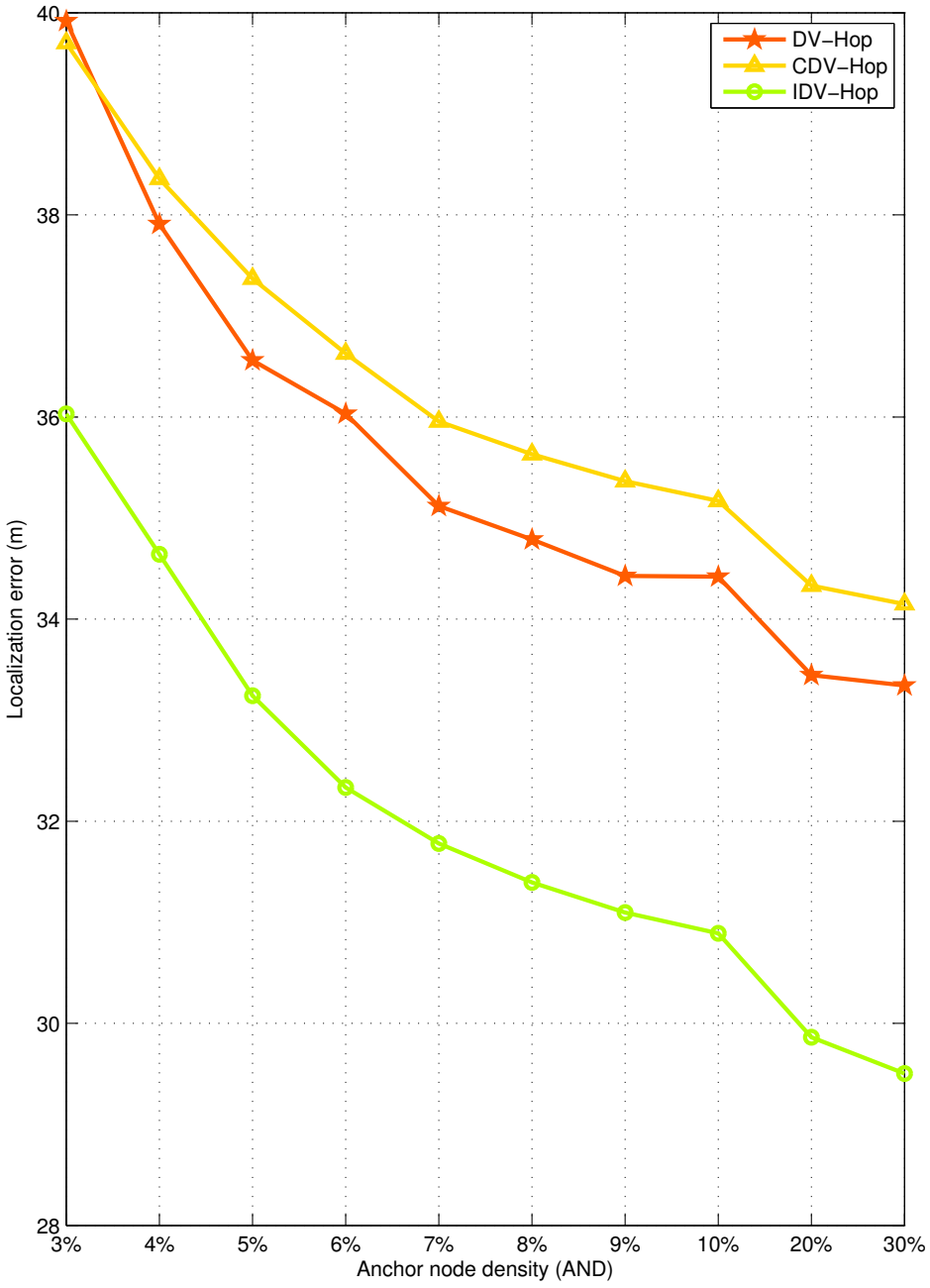

Figure 4.8: Comparison of localization error of DV-Hop, CDV-Hop and IDV-Hop when transmission radius is 105 m and anchor node density varies.

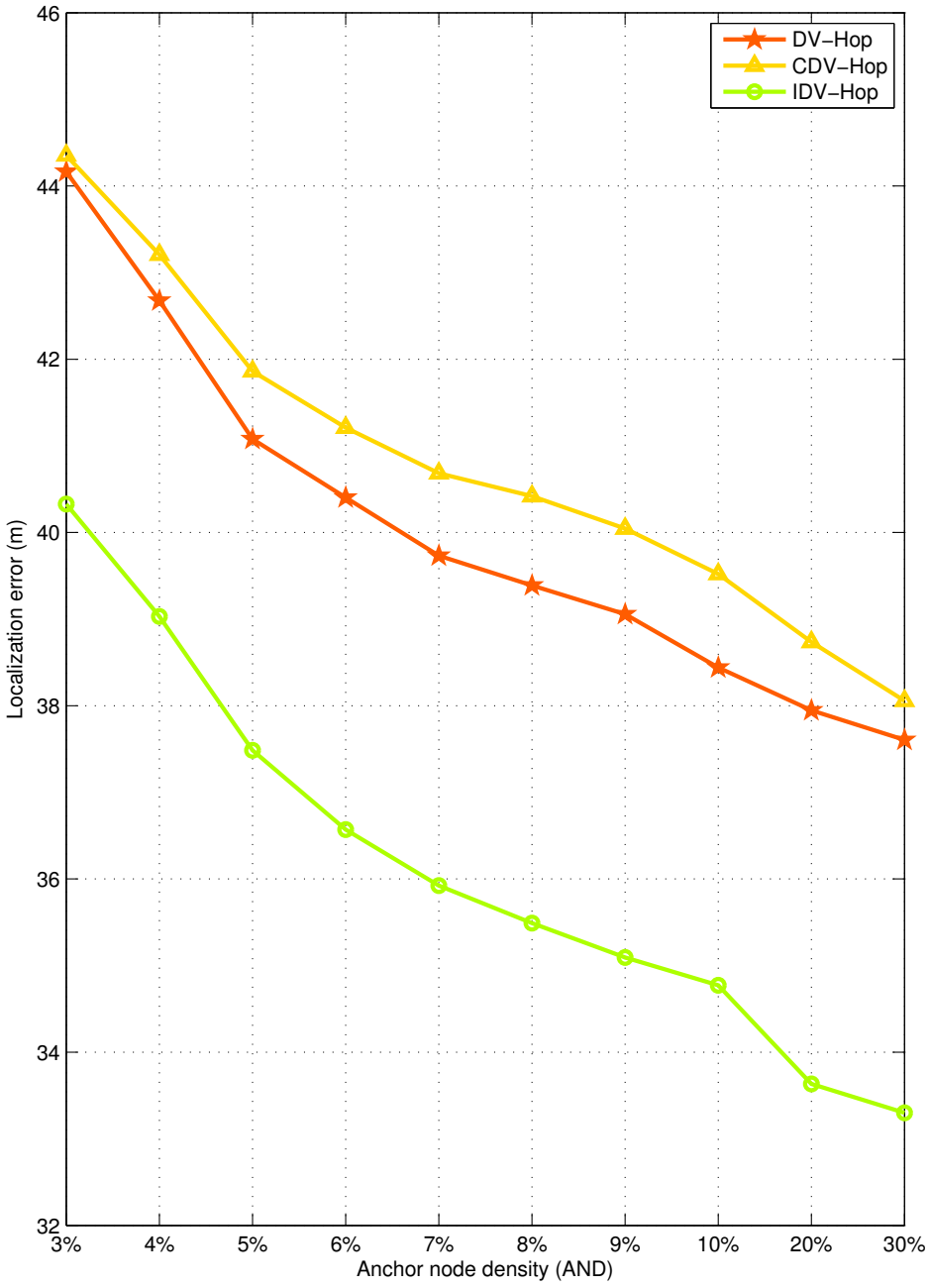

Figure 4.9: Comparison of localization error of DV-Hop, CDV-Hop and IDV-Hop when transmission radius is 120 m and anchor node density varies.

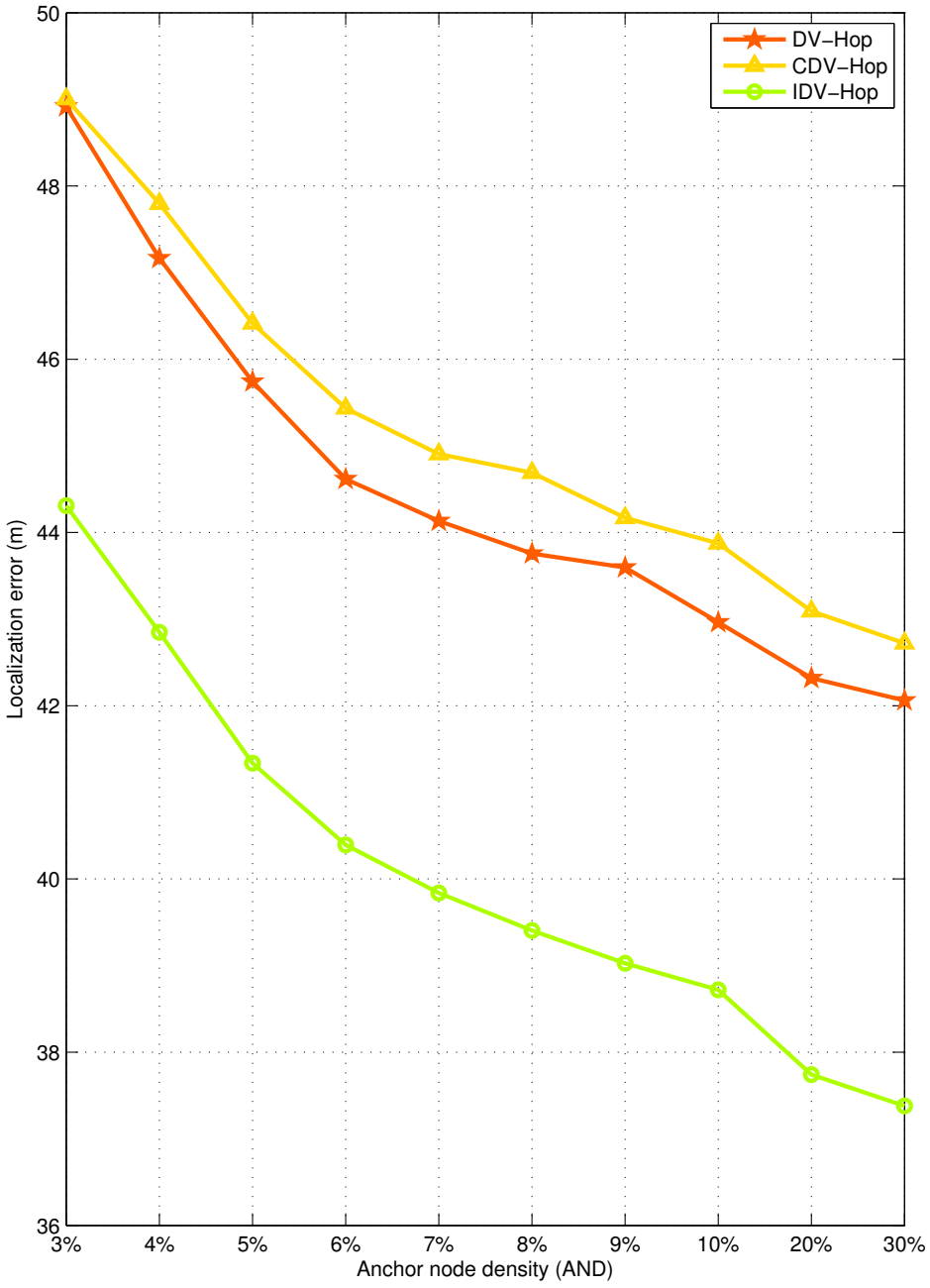

Figure 4.10: Comparison of localization error of DV-Hop, CDV-Hop and IDV-Hop when transmission radius is 135 m and anchor node density varies.

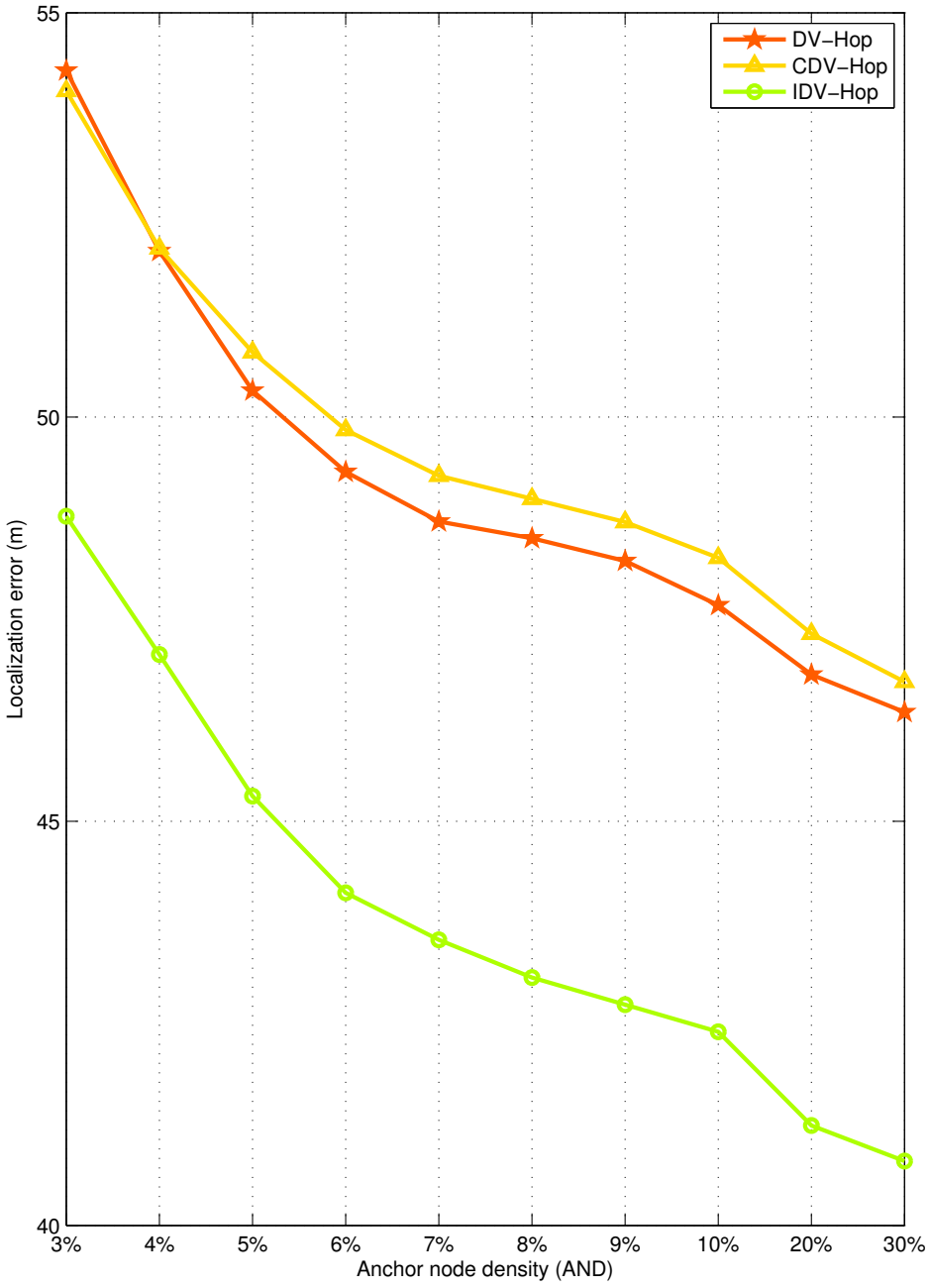

Figure 4.11: Comparison of localization error of DV-Hop, CDV-Hop and IDV-Hop when transmission radius is 150 m and anchor node density varies.

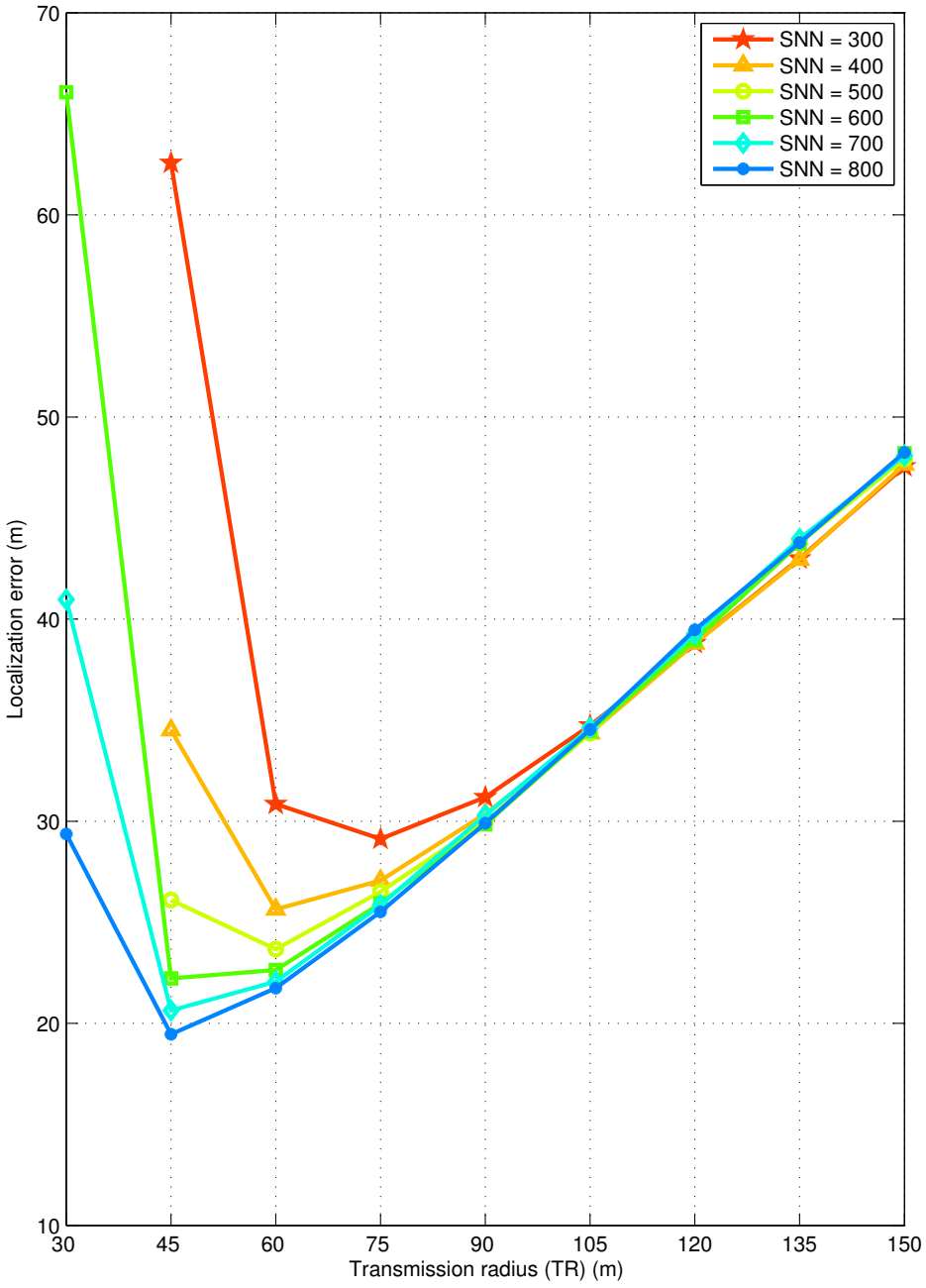

Figure 4.12: Localization error of DV-Hop versus transmission radius and number of sensor nodes.

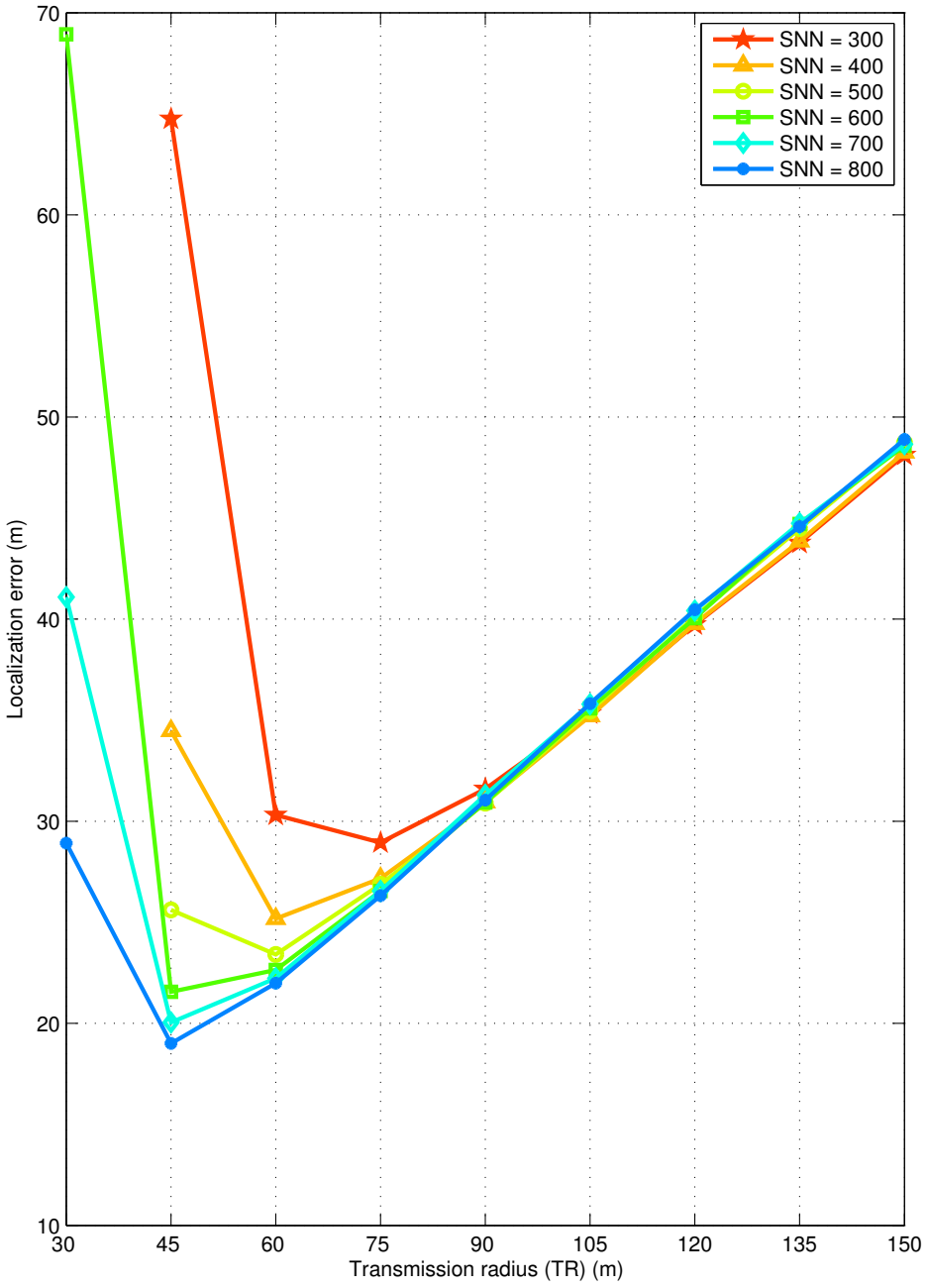

Figure 4.13: Localization error of CDV-Hop versus transmission radius and number of sensor nodes.

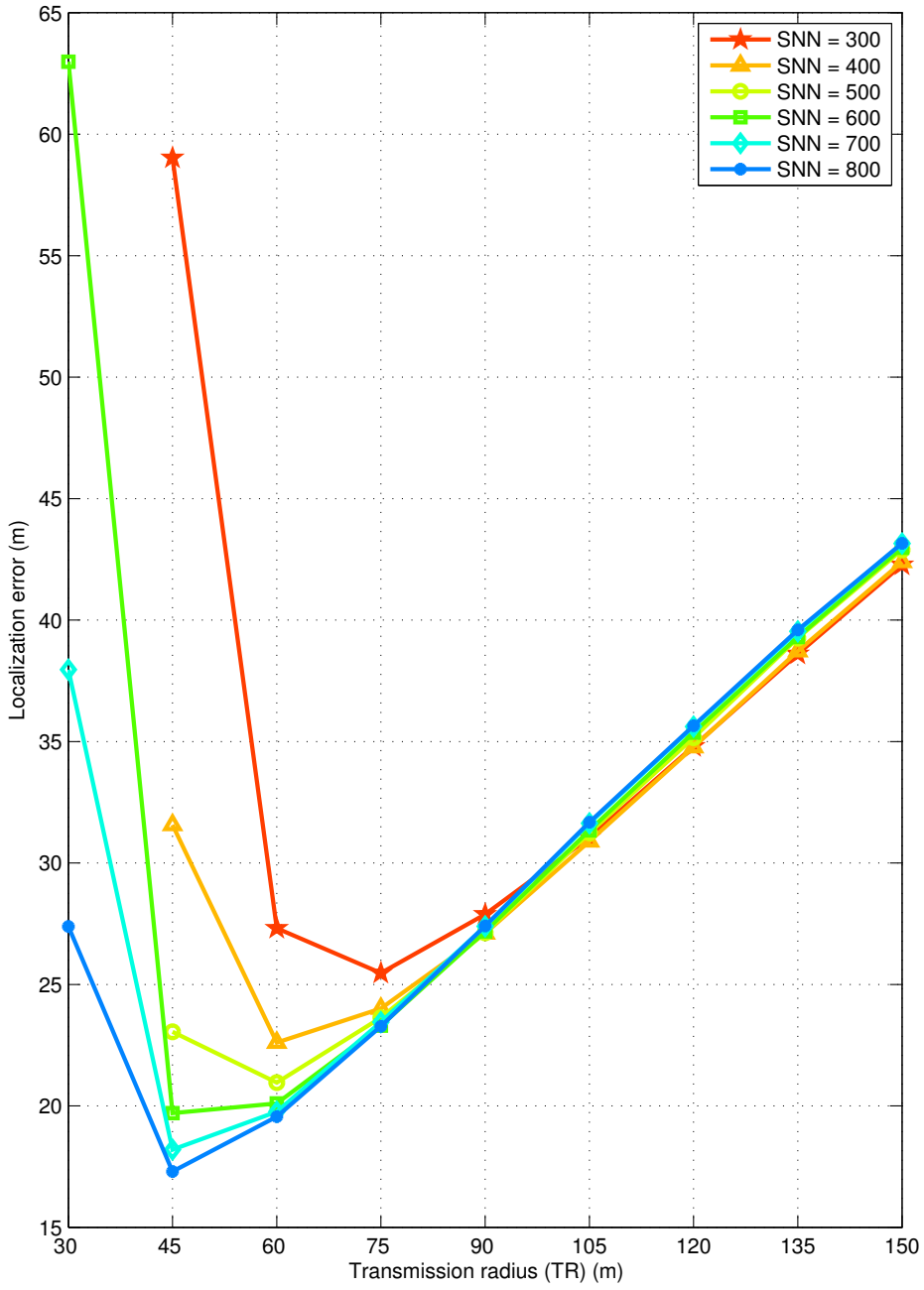

Figure 4.14: Localization error of IDV-Hop versus transmission radius and number of sensor nodes.

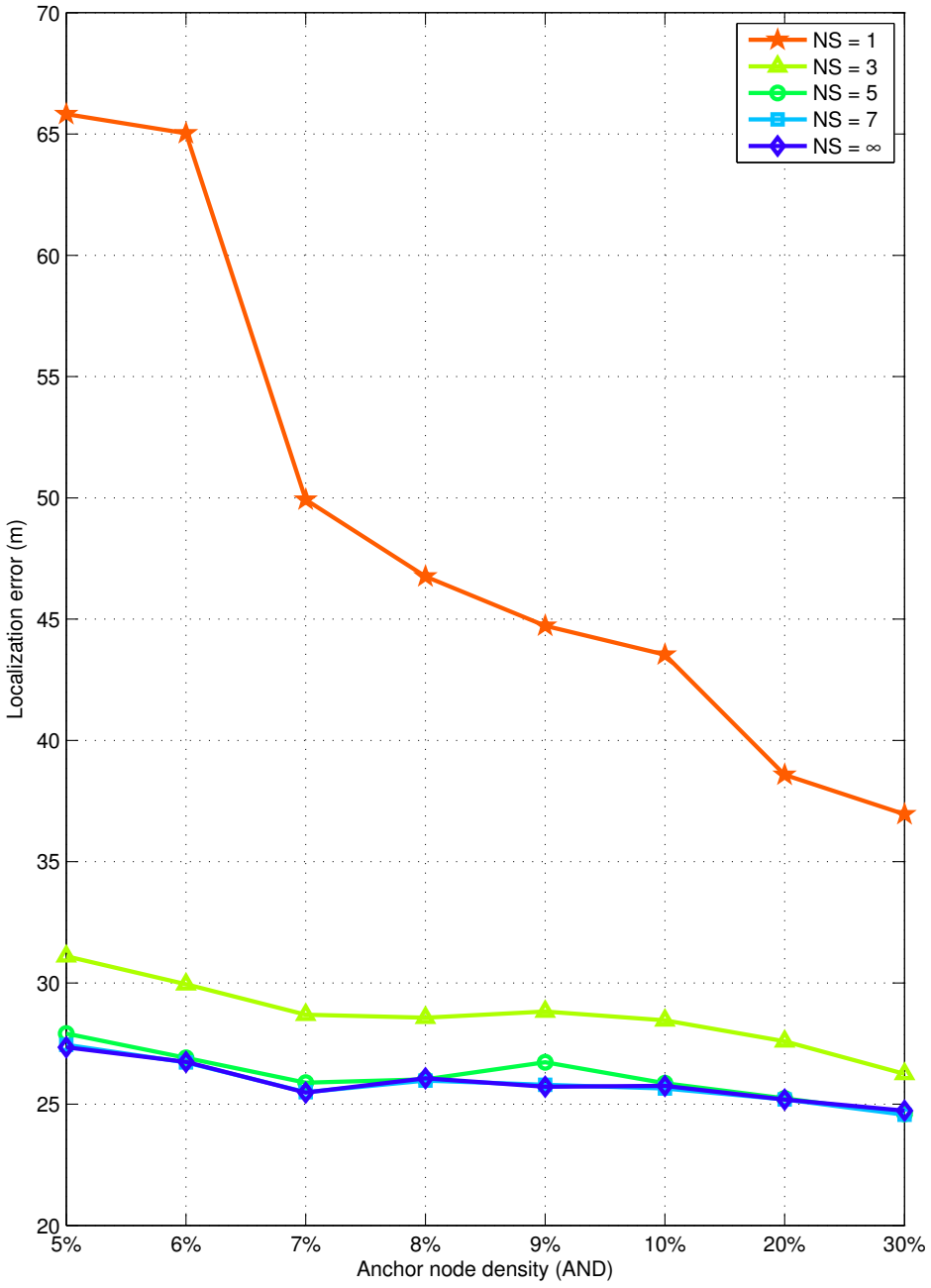

Figure 4.15: Localization error of DV-Hop versus neighborhood size and anchor node density.

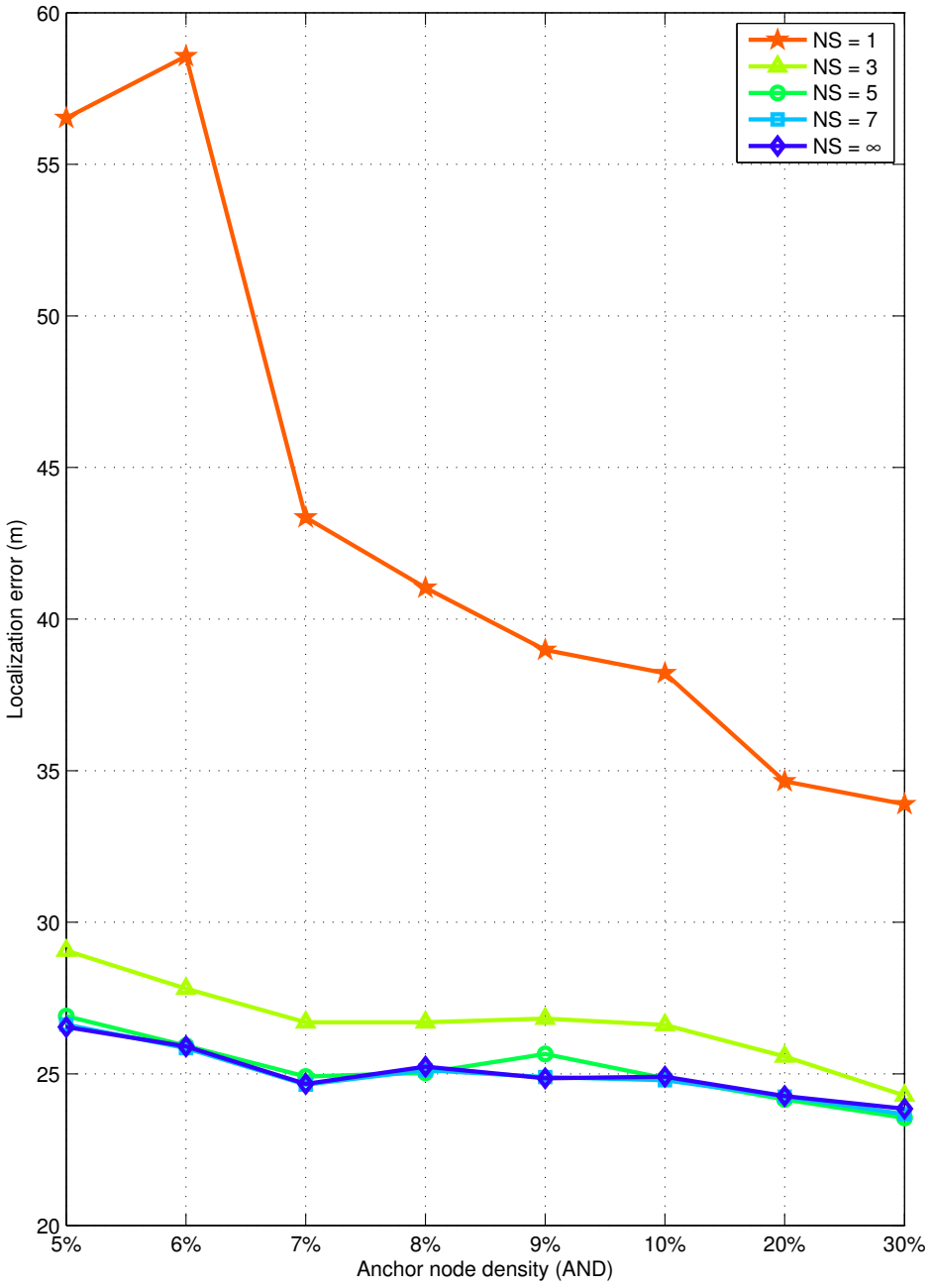

Figure 4.16: Localization error of CDV-Hop versus neighborhood size and anchor node density.

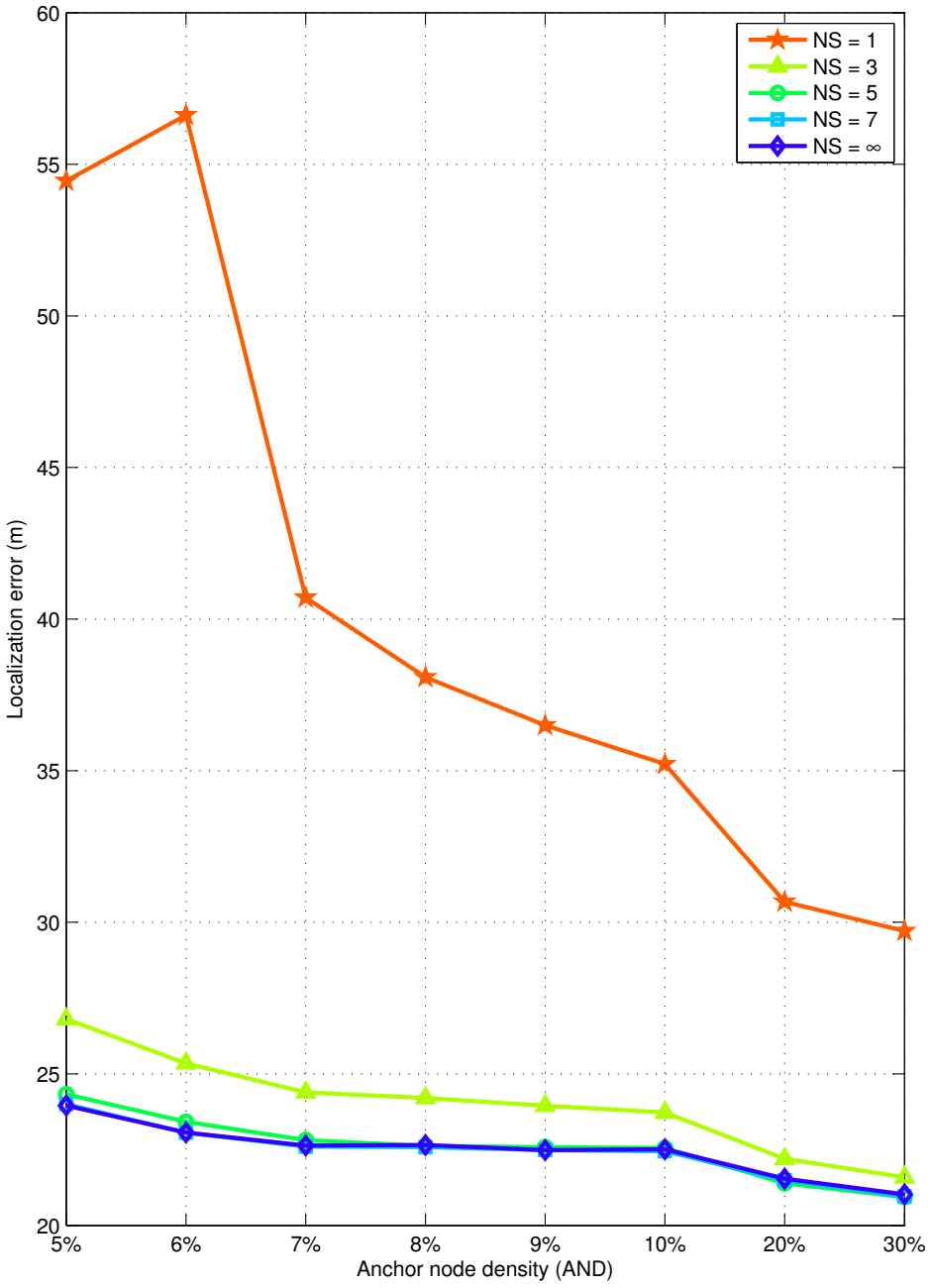

Figure 4.17: Localization error of IDV-Hop versus neighborhood size and anchor node density.

Chapter 5

Conclusion

This project evaluated three range-free localization algorithms for wireless sensor networks (WSNs): distance vector-hop (DV-Hop), DV-Hop with correction (CDV-Hop) and improved DV-Hop (IDV-Hop) in the Java programming language using an open source WSN simulator called NetTopo. The performance of the three DV-Hop algorithms was investigated in three experiments.

Experiment 1 investigated the impact of the anchor node density (AND) and the transmission radius (TR) of sensor nodes on the accuracy of the three algorithms. From Experiment 1, IDV-Hop achieved an average of 10% less localization error than DV-Hop. The performance of CDV-Hop did not show a conclusive pattern compared with DV-Hop. Given a TR, all three algorithms showed that an increase in the AND decreased the localization error. Given an AND, all three algorithms showed that 60 m produced the smallest localization error, hence was the best TR in the experiment. The non-monotonicity between the localization error and the TR was explained by the fact that an increase in the TR increases the quantization error after the WSN is fully connected.

Experiment 2 investigated the impact of the number of sensor nodes (SNN) and the TR. Given a set of TRs, all three algorithms showed that an increase in the SNN decreased the best TR.

Experiment 3 investigated the impact of the AND and the neighborhood size (NS) on the accuracy of the three algorithms with the connected k-neighborhood (CKN) algorithm. Given an AND, an increase in the NS decreased the localization error when the NS was in the range of 1 and 5. The localization error was close when the NS's were 5, 7 and ∞. For all three algorithms, 3 was the best NS in the experiment, considering both the localization error and the running time. For all

three algorithms, when the NS increased from 3 to 5, the average number of awake sensor nodes increased by 34%, but the decrease in the localization error were in the range of 6% and 10%. According to the computational complexity of the estimation of the coordinates, the estimated running times of all three algorithms increased significantly.

The best TRs in this project were selected from a set of values. In the future, optimization techniques can be applied to find a more precise value of best TR, given a size of WSN and an SNN. This project assumed the radio ranges of sensor nodes were perfect circles. In reality, there is packet loss between sensor nodes within the TR. The loss can be caused by background noise, fading of signals, signal interference, etc. Therefore, radio range irregularity models can be considered in the evaluation of the algorithms.

Bibliography

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "A survey on sensor networks", *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002

[2] G. Han, H. Xu, T. Q. Duong, J. Jiang and T. Hara, "Localization algorithms of wireless sensor networks: A survey", *Telecommunication Systems*, vol. 52, no. 4, pp. 2419–2436, 2013

[3] L. Girod and D. Estrin, "Robust range estimation using acoustic and multimodal sensing", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1312–1320, 2001

[4] A. Savvides, C. C. Han and M. B. Strivastava "Dynamic fine-grained localization in ad-hoc networks of sensors", *Proceedings of the International Conference on Mobile Computing and Networking*, pp. 166–179, 2001

[5] D. J. Torrieri, "Statistical Theory of Passive Location Systems", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 20, no. 2, pp. 183–198, 1984

[6] T. S. Rappaport, *Wireless Communications: Principles and Practice*, Upper Saddle River, NJ: Prentice Hall, 1996

[7] P. M. Djurić and S. M. Kay, "Parameter estimation of chirp signals", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, no. 12, pp. 2118–2126, 1990

[8] I. Stojmenovic (Ed.), *Handbook of Sensor Networks: Algorithms and Architectures*, Hoboken, NJ: John Wiley & Sons, 2005

[9] S. Čapkun, M. Hamdi and J. P. Hubaux, "GPS-free positioning in mobile ad hoc networks", *Cluster Computing*, vol. 5, no. 2, pp. 157–167, 2002

[10] D. Niculescu and B. Nath, "Ad hoc positioning system (APS)", *IEEE Global Telecommunications Conference*, pp. 2926–2931, 2001

[11] R. Nagpal, H. Shrobe and J. Bachrach, "Organizing a global coordinate system from local information on an ad hoc sensor network", *Information Processing in Sensor Networks*, pp. 333–348, 2003

[12] T. He, C. Huang, B. M. Blum, J. A. Stankovic and T. Abdelzaher, "Range-free localization schemes for large scale sensor networks", *Proceedings of the International Conference on Mobile Computing and Networking*, pp 81–95, 2003

[13] W. Yu and H. Li, "An improved DV-Hop localization method in wireless sensor networks", *International Conference on Computer Science and Automation Engineering*, pp. 199–202, 2012

[14] H. Chen, K. Sezaki, P. Deng and H. Cheung So, "An improved DV-Hop localization algorithm for wireless sensor networks", *IEEE Conference on Industrial Electronics and Applications*, pp. 1557–1561, 2008

[15] S. Nath and P. B. Gibbons, "Communicating via fireflies: Geographic routing on duty-cycled sensors", *International Symposium on Information Processing in Sensor Networks*, pp. 440–449, 2007

[16] TCS-Sensor Lab, 2016. *AlgoSenSim - Overview* [Online]. Available: `http://tcs.unige.ch/doku.php/code/algosensim/overview`

[17] Distributed Computing Group, ETH Zûrich, 2016. *Sinalgo - Simulator for Network Algorithms* [Online]. Available: `http://dcg.ethz.ch/projects/sinalgo`

[18] S. P. Fekete, A. Kroller, S. Fischer and D. Pfisterer, "Shawn: The fast, highly customizable sensor network simulator", *International Conference on Networked Sensing Systems*, pp. 299-299, 2007

[19] L. Shu, C. Wu, Y. Zhang, J. Chen, L. Wang and M. Hauswirth, "NetTopo: Beyond simulator and visualizer for wireless sensor networks", *International Conference on Future Generation Communication and Networking*, pp. 17–20, 2008

[20] NS-2, 2016. *NS-2 Wiki* [Online]. `http://nsnam.sourceforge.net/wiki/index.php/Main_Page`

[21] S. Park, A. Savvides and M. B. Srivastava, "SensorSim: A simulation framework for sensor networks", *Proceedings of the ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 104–111, 2000

[22] J. A. Miller, R. S. Nair, Z. Zhang and H. Zhao, "JSIM: A Java-based simulation and animation environment", *Proceedings of the Simulation Symposium*, pp. 31–42, 1997

[23] X. Zeng, R. Bagrodia and M. Gerla, "GloMoSim: A Library for parallel simulation of large-scale wireless networks", *Proceedings of the Workshop on Parallel and Distributed Simulation*, pp. 154-161, 1998

[24] P. Levis, N. Lee, M. Welsh and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications", *Proceedings of the International Conference on Embedded Networked Sensor Systems*, pp. 126–137, 2003

[25] D. Blazakis, J. McGee, D. Rusk and J. S. Baras, "ATEMU: A fine-grained sensor network simulator", *IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, pp. 145–152, 2004

[26] B. L. Titzer, D. K. Lee and J. Palsberg, "Avrora: Scalable sensor network simulation with precise timing", *International Symposium on Information Processing in Sensor Networks*, pp. 477–482, 2005