KNN Query Processing in

Wireless Sensor and Robot Networks

Wei Xie

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Master of Electrical and Computer Engineering degree

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering (SITE)
Faculty of Electrical and Computer Engineering
University of Ottawa
Ottawa, Ontario, Canada K1N 6N5

Acknowledgements

I would like to acknowledge my sincerest gratitude to my supervisor Prof. Dr. Ivan Stojmenovic of Ottawa-Carleton Institute for Computer Science. Without his enthusiastic and expert guidance during writing this thesis, it would not have been possible to complete this thesis. I am grateful to my co-supervisor Prof. Dr. Amiya Nayak for the help in preparation of this thesis. I would also like to thank my friends and my family in particular for their continuous understanding and support.

Table of Contents

List of Figures

Abstract

In Wireless Sensor and Robot Networks (WSRNs), static sensors report event information to one of the robots. In the *k* nearest neighbour query processing problem in WSRNs, the robot receives event report needs to find exact *k* nearest robots (KNN) to react to the event, among those connected to it. We are interested in localized solutions, which avoid message flooding to the whole network. Several existing methods restrict the search within a predetermined boundary. Some network density-based estimation algorithms were proposed but they either result in large message transmission or require the density information of the whole network in advance which is complex to implement and lacks robustness. Algorithms with tree structures lead to the excessive energy consumption and large latency caused by structural construction. Itinerary based approaches generate large latency or unsatisfactory accuracy. In this thesis, we propose a new method to estimate a search boundary, which is a circle centred at the query point. Two algorithms are presented to disseminate the message to robots of interest and aggregate their data (e.g. the distance to query point). *Multiple Auction Aggregation* (MAA) is an algorithm based on auction protocol, with multiple copies of query message being disseminated into the network to get the best bidding from each robot. *Partial Depth First Search* (PDFS) attempts to traverse all the robots of interest with a query message to

gather the data by depth first search. This thesis also optimizes a traditional itinerary-based KNN query processing method called IKNN and compares this algorithm with our proposed MAA and PDFS algorithms. The experimental results followed indicate that the overall performance of MAA and PDFS outweighs IKNN in WSRNs.

Chapter 1: Introduction

1.1. Background Information

Wireless Sensor Network (WSNs) typically consists of a large number of resource-constrained sensing devices in tiny size, which are powered by low-energy batteries and connected via wireless communication links. The basic function of the sensors is to monitor environmental conditions (e.g. the temperature, the sound etc.) and to report the data to a sink and the sink will react to the data or forward it to further terminals.

Multi-Robots Systems (MRS) is an important branch of the research in Robotics due to multiple potential applications on areas such as forest fire detection, transport systems etc. Recently, several works aiming at studying the coordination between WSNs and MRS received significant attention. This new area which is called Wireless Sensor and Robot Networks (WSRNs) provides the prospect for robots to optimize the performance of WSNs, such as sensor deployment [Chang et al. 2009] and sensor relocation [Wang et al. 2005]. Another prospective application of WSRNs is to enable robots to interact with sensors immediately after receiving the message from sensors, such as serving as aids for fighting forest fires or reacting to emergency situations sensed by

sensors [Kumar et al. 2004]. An example of WSRNs is illustrated in Fig. 1. In this graph, red circles stand for robots and the black dots stand for sensors.

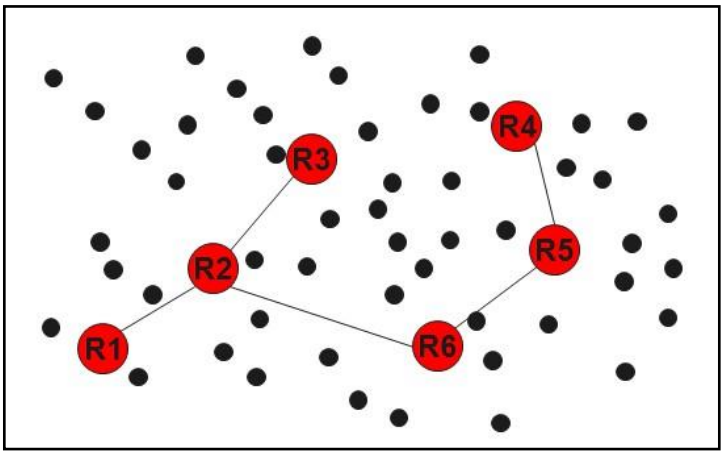


Fig 1 Wireless and Robot Network

Recent research has proposed multiple algorithms to KNN query processing problem with various structures. KNN query problem in centralized system which stores all the information of sensors and robots in a database center received the attention of a number of researchers [Roussopoulos et al. 1995, Song and Roussopoulos, 2001]. However, this structure is not recommended in WSRNs due to the complex index structure, redundant transmission and high energy consumption. Thus, most of the recent researches focus on processing KNN query in a localized system which indicates each node only knows the information about its neighbour and itself. Existing localized based KNN query processing techniques can be classified into two categories: infrastructure based technique and infrastructure free technique. The former relies on

network infrastructure such as the tree structure. The latter does not rely on any infrastructure and most solutions of this type are itinerary based approaches [Chempavathy and Vijayaraja, 2010, Jayaraman et al. 2010, Wu et al. 2008, Wu et al. 2007 and Winter et al. 2005] which will be introduced in Chapter 2.

In this thesis, we will focus on the localized solutions in KNN query processing problem in WSNs and WSRNs and propose several approaches. We will also examine previous research on both infrastructure-based and infrastructure -free algorithms and explore their merits and drawbacks.

1.2. Problem Statement

To better understand our problem in Wireless Sensor and Robot Network, we first review the KNN query processing problem in Wireless Sensor Network. As previously stated, one of the most essential problems in WSNs is KNN (*k* Nearest Neighbours) query processing which aims at searching for *k* nearest neighbours of the query point and sorting them by their Euclidean distance. Theoretically, KNN query processing problem can be defined as following:

Definition (*k* Nearest Neighbour problem): *Given a set of nodes (sensors or robots) N, a geographical location q (denoted by a query point q) and valid time T, find N' with k objects ($N' \subseteq N$, $|N'|=k$) that* $\forall n_1 \in N'$. *At time T:*

$$\forall n_1 \in N', n_2 \in N - N' : D(n_1, q) < D(n_2, q).$$

Where D denotes the Euclidean distance function.

While in WSRNs, Generally speaking, there are two new types of coordination: sensor-robot and robot-robot coordination. Sensor-robot coordination provides information about the event sensed by sensors and the message transmission is from sensors to robots. Having received event information, robot-robot coordination provides solutions for robots to make a decision on how to act upon the event. We will mainly focus on robot-robot coordination in this thesis. In robot-robot coordination, for the robustness and reliability concern, multiple robots should be selected as a task team instead of one single robot. Thus, the task turns to a typical KNN query processing problem, which implements data dissemination and aggregation via query message forwarded to the robots located in a geographical proximity specified by a given query point and a sample size *k*. Then the list of *k* nearest robots is obtained.

Specifically, in this thesis, KNN query processing problem in WSRNs is defined as follows. In WSRNs, sensors and robots are deployed randomly. An event occurs and is sensed by a sensor, and this sensor tries to find *k* robots nearest to the query point, sorting them by their Euclidean distance. In this thesis, we assume that a query message aiming at searching for *k* nearest robots to act upon the sensed event is generated by this sensor. Afterwards, the query message is forwarded over the sensor network to a robot in the vicinity (denoted as Vicinity

Robot), which may not be the closest one to the query point. This robot then forwards the query point via routing algorithm to the robot (denoted as the Home Robot) nearest to the query point over robot network. The problem is how to disseminate the query message to the robot network efficiently and find *k* nearest robots with high accuracy and low latency. A demonstration of our problem statement in this thesis is shown in Fig. 2. As illustrated, an event is sensed by a sensor and this sensor forwards the query message to a robot in the vicinity (R4). The query message is forwarded by R4 in the robot layer and eventually reaches the robot nearest to the query point. The message routing in sensor layer is marked with solid arrows while the message routing in robot layer is presented with dashed arrows.

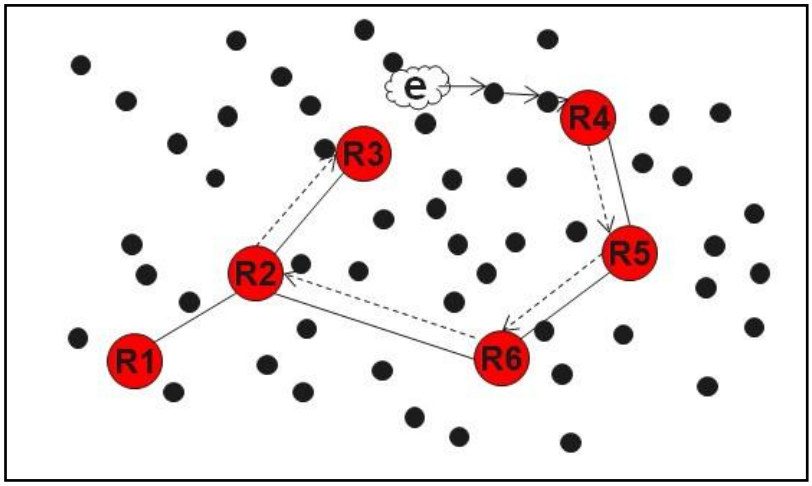


Fig. 2 KNN Query Process in WSRNs

In this thesis, we call an algorithm as *Guarenteed Delivery* if they could guarantee that exactly *k* nearest robots are found at the end of the

algorithm. However, to the best of our knowledge, all the existing solutions for KNN query processing do not guarantee a solution to find the exactly *k* nearest robots. That is, it finds *k* robots near the event point, but not precisely *k* nearest ones. Under such assumption, the accuracy of the solution should be considered as a parameter to evaluate an algorithm.

Some assumptions should be clarified for ease of understanding. In this thesis, the sensor network and the robot network operate with different responsibilities. The occurrence of an event can be only sensed in the sensor network. After the query message enters the robot network (i.e. the query message is received by the robot), successive process (i.e. query message dissemination and data aggregation) will be only operated in robot network. The scenario that robots communicate with each other via the sensor layer is expected, but we will not take this method into consideration for energy efficiency and stability reasons. Specifically, due to the limited size, sensors are expected to operate within an extremely frugal energy budget, and limited number of message transmissions is expected before the batteries are drained. Also, sensor nodes have limited storage, communication and computation abilities. While in robot layer, robots are considered to be resource-rich as compared to sensors, with better hardware capabilities (e.g. battery, storage room, antenna etc.). As a result, in KNN query process, we should assign the tasks of message propagation and data processing to the robot and enable the sensor to save resource for monitoring. This

assignment of responsibilities enhances the efficiency and sustainability of WSRNs.

The basic execution phases of our approach (KNN query processing for WSRNs) and the previously stated approach (KNN query processing for WSNs) are similar. The query message is sensed by sensor and a KNN boundary is generated to restrict the searching. The main difference between these two is that in KNN query processing for WSNs, the query is forwarded to the nearest sensor around the query point and the subsequent message dissemination is operated over the sensor network. While in KNN query processing for WSRNs, the query message is firstly forwarded to a robot in the vicinity and this robot will forward this message to the nearest robot instead of a sensor around the query point. Subsequent message dissemination is operated over the robot network. Considering the assumption that the query message in WSRNs will route over robot network and no sensor is further involved after the Home Robot receives the query message, the solutions applied in WSNs to solve KNN query processing are also valid for our solution. The only difference is that the carrier of the message in WSRNs is the robots, corresponding to the sensors in WSNs. This is also the reason we present them in Chapter 2.

1.3. Existing Solutions

Most recent works focused on searching for *k* nearest sensors in WSNs. In this thesis, however, we pay attention to searching for *k* nearest robots to react to the query sensed by sensors within a given boundary in WSRNs. However, the algorithm to execute KNN query processing problem in WSNs can be applied to solve our algorithm. Generally speaking, KNN query processing consists of three phases. We will briefly introduce them in sequence:

- Routing and Home Node Search phase: Event is sensed by senor and a query message is generated and forwarded to the nearest node (Denoted as the Home Node).
- KNN Boundary Estimation phase: The Home Node estimates a KNN boundary centred at query point with a radius of *r* to restrict the message propagation.
- Query Dissemination and Data Aggregation phase: Query message is disseminated into the network to aggregate the data of nodes of interest and then forwarded back to the Home Node for processing.

1.3.1. Home Node Search Phase

The most commonly used algorithm in this phase is the GPSR algorithm presented by Karp and Kung (2000). This algorithm works in two modes: Greedy Mode and Perimeter Mode. The message is

forwarded by GPSR in Greedy Mode and enters Perimeter Mode once Greedy fails. Then the message is forwarded by left hand or right hand rule until Greedy Mode is available again. This is a guaranteed message delivery algorithm. However, this algorithm requires every node to store the information about its neighbours.

Komai et al. (2011) presented a beacon less algorithm. This algorithm does not require the current node to store its neighbour's information. However, this algorithm is proved to be not a guaranteed algorithm.

Denoting the source point as Vicinity Robot and the destination point as query point, both of these two algorithms can be applied to forward the query message to the Home Node.

1.3.2. KNN Boundary Estimation Phase

Winter and Lee (2004) proposed the Maximum Hop distance (MHD) algorithm to calculate KNN boundary. This approach generates a counter variable at the beginning of the Home Node search phase to record the largest hop distance between the source and Home Node. The variable is kept in the query message and updated only if a larger distance between two nodes is found. When the message reaches the Home Node, the maximum KNN boundary can be estimated by multiplying MHD value by *k*. This approach requires small storage space and operates with low protocol complexity. However, the drawback is that the accuracy may be

very low when one of the hop distances collected by this progress is extremely large and recorded as MHD.

Several studies on density-based boundary estimation approaches have been conducted. Fu et al. (2010) presented an approach to estimate node density by linear regression technique. The basic idea of this approach is to estimate KNN boundary based on the data collected in Home Node Search phase. In Home Node Search phase, multiple robots are visited before the query message reaches Home Node and their data indicates the density of the network. Thus, the author presented a linear regressive equation to estimate the boundary. Jayaraman et al. (2010) proposed another approach to estimate KNN boundary in 3D sensor network. In this approach, the density of the entire network is calculated and stored in each node in advance, thus the expectation of r to ensure that there are k nodes within the circle with radius r can be estimated based on the density. However, the prior approach requires data aggregation in every hop during the Home Node Search phase which consumes extra energy and generates large latency. The latter algorithm needs to store the density of the network in nodes in advance which is complex to implement and lack of robustness.

1.3.3. Query Dissemination Phase

The most essential challenge in KNN query process is how to disseminate the query message into network and aggregate the data of nodes of interest. Generally, the previous studies on KNN query processing can be classified into two categories: centralized approach and localized approach. As stated in Chapter 1.2, centralized approach is not a satisfactory solution to KNN query processing problem. Thus, most recent researches are focusing on localized solutions. Localized approaches can be further divided into two sub-categories: infrastructure-based approaches which require structure construction before message routing, and infrastructure-free approaches which do not rely on certain network structures.

Most infrastructure-based approaches are achieved by tree structure. Demirbas and Ferhatosmanoglu (2003) presented the Peer Tree structure to address KNN query. In this approach, the search region is divided into a hierarchy of Minimum Bounding Rectangles (MBR) and every node in each MBR will report its data to one cluster head node and by this, it means that data is collected. Winter and Lee (2004) presented KNN Perimeter Tree (KPT) algorithm to search the locations and IDs of the rest of nodes. The basic idea of this infrastructure approach is to divide the region within the boundary circle into small sub regions and to construct minimum spanning trees in the direction away from the destination. However, the drawbacks of these infrastructure-based

algorithms are an excessive energy consumption and large latency caused by structural construction.

Recently, a number of itinerary-based algorithms are proposed to provide the solution to infrastructure-free approaches. The basic idea of an itinerary-based algorithm is to enforce each node which receives the query message to forward the message to the next target node over a predetermined itinerary. Before forwarding the query message, the node inserts the information of its neighbour and itself into the query message. When the query message reaches the KNN boundary, the result is forwarded back to the Home Robot for processing. The principal difference between existing itinerary-based solutions is the format of the itineraries. Itinerary-based KNN query processing (IKNN) (Xu et al. 2007) proposed an Archimedean Spiral itinerary, and studied the Parallel Itinerary. Density-aware Itinerary-based KNN query processing (DIKNN) (Wu et al. 2007) divides the region encircled by KNN Boundary centralized at query point into cone-shapes and generates itinerary in each sub-region. PIKNN (Fu et al. 2010) also partitions the search region and generates the parallel concentric-circle itineraries. PIKNN enables more KNN query threads than the number of itinerary to be generated during the progress of propagation. The drawback of itinerary based algorithms is that the latency may be significantly increased with long itinerary. Besides, *Itinerary Void* (Xu et al. 2006) may occur which lowers the accuracy of algorithms.

1.4. Motivations and Objectives

As an alternative to existing KNN boundary estimation algorithm, we expect a more efficient and accurate approach. This algorithm should estimate the partial network density around the query point instead of the average value of the entire network. To improve the accuracy and robustness, instead of one, multiple samples should be involved during the processing. Following accuracy, the latency should be considered as a second criterion to evaluate the estimation.

Although extensive works have been done to solve KNN query processing in WSNs, very few researches are conducted to solve the KNN query processing problem in Wireless Sensor and Robot Network (WSRNs), and this is the main motivation of this thesis. In WSRNs, robots, instead of sensors, undertake most work of message transmission and processing. Due to the factor that robots are considered to be resource-rich as compared to sensors with better hardware capabilities (e.g. larger batteries), energy consumption becomes a less significant criterion in WSRNs. In WSRNs, we aim at exploring a query dissemination and data aggregation algorithm with high accuracy, low latency and acceptable energy consumption.

Motivated by these ideas, in Chapter 3, we propose one KNN boundary estimation algorithm and three possible algorithms to implement KNN query processing.

1.5. Assumptions

The concept of the WSNs in this thesis will consist of homogeneous sensors. These sensors are assumed to be static. The communication range is twice as large as sensing range and is the same for all sensors. Any two sensors may communicate with each other directly if their separation is less than the communication ranges. As a result, all links are assumed to be bi-directional. All the sensors in WSNs are deployed randomly and no isolated sensor (i.e. the degree of sensor is 0) exists.

WSRNs, as an extension of WSNs, are assumed to maintain all the features and assumptions of WSN. Besides, one or more robots are deployed randomly in the WSN network. These robots are supposed to be more resource-rich as compared to sensors. Robots can communicate with each other (i.e. communicating in robot layer). The occurrence of an event can be only sensed in sensor layer. After the query message enters the robot layer (i.e. the query message is received by the robot), successive process (i.e. query message dissemination and data aggregation) will be only operated in robot layer.

We also assume that the occurrence of the event is always sensed by a sensor in the sensor layer. The WSRNs in this thesis is always capable of performing only one task at a time, which means only one query event is assumed to exist in WSRNs. The subsequent event will not occur until the result of the current event is obtained.

Ideal physical and MAC layer are used. Message transmissions have no collisions and no losses. Message transmission within the WSRNs is assumed to be asynchronous. Therefore, all messages will eventually reach the desired destination. However, several fault-tolerant schemes will be discussed to optimize the algorithms.

1.6. Contributions

We present a novel approach to estimate KNN Boundary. After receiving query message, Home robot will send a probe message to all its neighbours. This message will only be forwarded to Home Robot's neighbours and should not be propagated further. Each robot receiving the query message should calculate the average distance between its neighbours and itself and then forward the probe message back to Home Robot. The Home Robot will calculate the average distance between its neighbours and itself (denoted as *d*), and then the KNN boundary can be estimated by the following equation:

$$r = kdc$$

Where *k* is the number of target robots and *c* is adjustable coefficient. The rest of Chapter 3.1 verifies the rationality of the estimation. This approach estimates the KNN Boundary based on multiple samples and indicates partial network density around query point. Also, only Home Robot needs to send the probe message to gather necessary data which enables the latency to maintain under a low level.

We proposed two algorithms to address KNN query process and these three algorithms are presented as follows.

- Multiple Auction Aggregation (MAA): Inspired by Auction protocol (Komai et al. 2011), MAA is an infrastructure-based algorithm. This multithread algorithm constructed a tree rooted at Home Robot and disseminated multiple copies of the query message into the network to get the best biddings (i.e. distance with query point) from the children and then sorted them by distance.
- Partial Depth-First Search (PDFS): With a low protocol complexity, PDFS approach attempted to traverse all the robots with Depth-First Search encircled by KNN boundary and aggregate their data for processing.

Several fault tolerant schemes for MAA and PDFS are discussed later to enhance their performance.

Through extensive simulation, we evaluate the performance of MAA, PDFS and a comparative algorithm: IKNN (Xu et al. 2007). Based on our experiment result, the accuracy of MAA and PDFS outperforms the accuracy of IKNN on almost all occasions. However, MAA and PDFS suffer larger energy consumption than IKNN especially when the KNN boundary is expanded or the network density is increased. Considering the fact that the message transmission is operated in robot network layer and robots always carry much larger energy source than sensors, this drawback is less significant in most occasions. The latency of PDFS is

generally larger than PDFS but the latency of MAA is relatively minimal compared to IKNN and thus offers a quick reflection in emergency occasions in WSRNs. To conclude, with up to 30% more accuracy and 50% reduction in query response time, MAA is proved to be a satisfactory algorithm. Even the latency and the energy consumption of PDFS is generally larger than IKNN, considering the contribution on improvement of accuracy (up to 30%), PDFS is a satisfactory approach.

1.7. Organization of the Thesis

The remainder of this thesis is organized as follows:

- Chapter 2: Provides a literature review on the three basic phases of KNN query processing.
- Chapter 3: Proposes how we execute this three phases in our approaches.
- Chapter 4: Presents a comparative algorithm and compares it with our approach.
- Chapter 5: Summarizes the whole thesis and proposes several works for the future.

Chapter 2: Literature Review

As stated before, the solutions to KNN query processing in WSN can be applied to solve KNN query processing problems in WSRNs. As a result, reviewing these approaches and ideas are important and necessary. In this chapter, we will review and analyse the approaches in WSN which contribute to our thesis, in sequence.

2.1. Home Node Search

2.1.1. GPSR Algorithm

The initial phase of KNN query processing algorithm is searching the geographically nearest neighbour to the query point. This node, which is defined as Home Node, is assigned as a temporary server. The Home Node takes the responsibility of initiating routing algorithm, gathering the data, processing the result and making decisions during one single KNN query processing.

Greedy Perimeter Stateless Routing (GPSR) algorithm is applied to search for Home Node in most of the current research works. The GPSR, which is first presented by Karp and Kung (2000) consists of two methods for forwarding packets: Greedy Mode, which is the main method of this algorithm and is used as much as possible to forward the packet, and Perimeter Mode, which is utilized when Greedy Mode fails. In Greedy Mode, the forwarding node forwards the message to its

geographically nearest neighbour of the destination node. If this method fails, that means the forwarding node itself is the nearest node to the destination. The node then switches to the recovery mode and initiates perimeter forwarding. This mode uses Right-hand or Left-hand rules to traversal message until the destination is reached or greedy mode is eligible again. By keeping running these two methods along with the trace, the message eventually arrives at the destination node.

In the research conducted by Fu et al. (2007), GPSR is applied in searching for Home Node. In this paper, an arbitrary node (called source node) issues a KNN message and forwards the message via a given routing protocol (e.g. GPSR) to a nearest node (Home Node). The destination point of GPSR is the query point. As a result, the destination can never be reached and a routing loop around query point with perimeter mode is created. By collecting the distance information in this mode, Home Node is eventually confirmed.

2.1.2. Beacon-Less Routing

Komai et al. (2011) demonstrated a new approach to implement KNN query processing. In this paper, the Beacon-Less Routing Algorithm (BLR) (Heissenbuttel et al. 2004) is presented to search for Home Node. This algorithm, which is introduced by Heissenbuttel et al., performs routing in distributed wireless network without storing the information about neighbour nodes. In BLR, a *Dynamic Forwarding*

Delay (DFG) is calculated in every node which intends to forward the message. The node with optimal position obtains the shortest delay and thus forwards the message first. After the relaying of this optimal node, other nodes within a predetermined forwarding area with larger delay realize the successful transmission of the package and then cancel the relay progress.

The forwarding area should be large enough to increase the probability of finding a relatively optimal node within this area. However, an overrated forwarding area may lead to unnecessary energy consumption and message collision. This paper proposed three shapes of possible areas called a sector, a Reuleaux triangle and a circle. With the analysis conducted by the author presented later, circle has been proved as the best shape among these three in terms of forwarding area. A circle forwarding area gives approximately same progress per hop compared to other two methods but maintains higher numbers of successful hop before basic relay mode fails.

The value of DFG is determined by multiple parameters. For a particular node, this approach denotes p as the progress towards the destination node, and d as the distance between the node and the line from the source node to destination node. With a predetermined parameter *Max_delay*, the DFG value *Add_delay* can be calculated within the interval [0, *Max_delay*] by p and d.

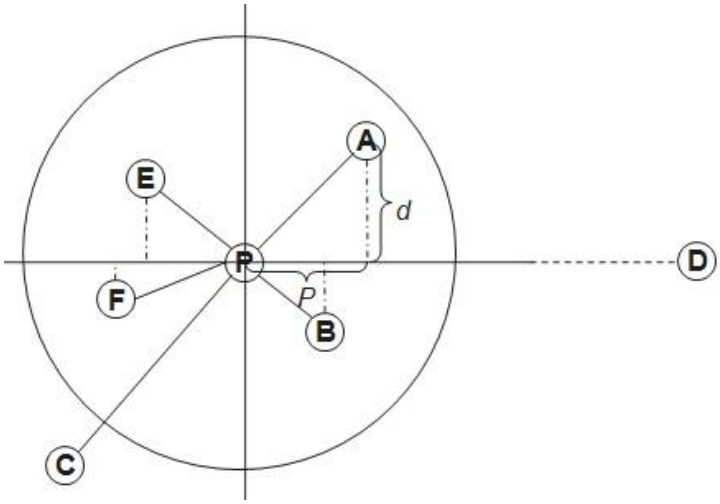


Fig. 3 BLR Algorithm

An example of BLR can be seen in Fig. 3. Node P intends to send the message to destination node D. Then it floods the message to every node within the forwarding area. As we can see from the graph, node C is out of the boundary and thus it cancels any forwarding progress after receiving the package and realizing it. Each node within the forwarding area (node A, B, E and F) calculates the DFG, respectively. Node A suffers least *Add_delay* and after the period of *Add_delay* it broadcasts a passive acknowledgement to other nodes within the forwarding area to inform them to cancel their scheduled transmission. Eventually, node A initiates another relay to continue the routing.

This approach substantially reduces the signalling overhead by partial flooding. However, an analysis conducted by Ian et al. clarified that this greedy routing is not a guaranteed delivery even though the path between the source node and the destination node exists (Akyildiz et al.

2005). Also, an experiment conducted by Sanchez et al. (2009) verified that BLR engenders an enormous number of duplicates and thus decreases the performance of the protocol to undesirable levels.

2.2. KNN Boundary Estimation

After Home Node being confirmed, we continue to find the rest of *k*-1 nodes. A naive way to achieve this goal is to simply flood the information to the whole network and gather information from every other node, and then sort them by distance. Obviously, this approach is inefficient and unnecessary. As a result, a search boundary is estimated and search progress is restricted in this area. Also, the precision of KNN boundary estimation has a significant impact on the KNN query processing. More unnecessary nodes will be involved if the boundary is overestimated, which will result in large energy consumption and latency. On the contrary, an underestimated boundary leads to the situation that less than *k* nodes is found at the end of the processing (i.e. $|N'|<k$). Home node then expands the boundary and executes routing algorithm again, which is also not energy saving.

2.2.1. Maximum Hop Distance

Winter and Lee (2004) proposed the Maximum Hop distance (MHD) algorithm. In this approach, a counter variable is generated at the beginning of Home Node Search phase to record the largest hop distance

on the route between the source and Home Node. This variable is kept in the package and will be updated only if a larger distance between two nodes is found. When the message reaches the Home Node, the maximum KNN Boundary can be estimated by multiplying MHD value by *k*.

Fig. 4 illustrates an example of boundary estimation by MHD. In this case, maximum hope distance can be confirmed as *d* after GPRS routing. Assume that $k = 2$, a circle with centre point *q* and a radius of $2d$ is determined as the KNN boundary.

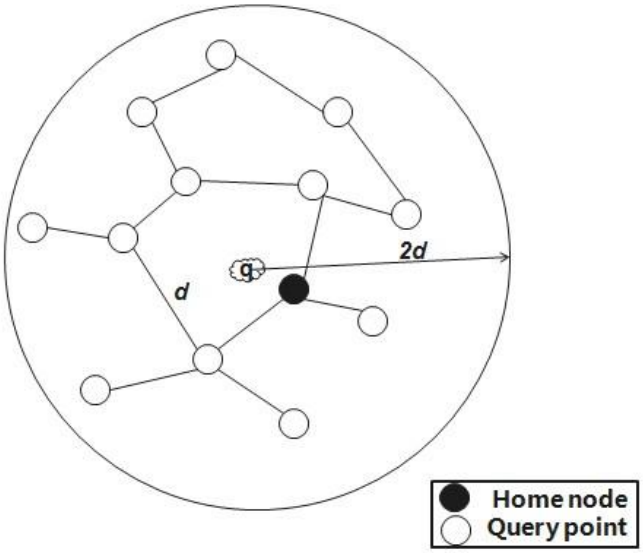


Fig. 4. MHD Boundary Estimation

However, the boundary obtained by this algorithm is likely to be a rough estimation. Particularly, the search boundary tends to be relatively large as *k* increases which leads to excessive energy consumption and long latency.

2.2.2. Network Density Based

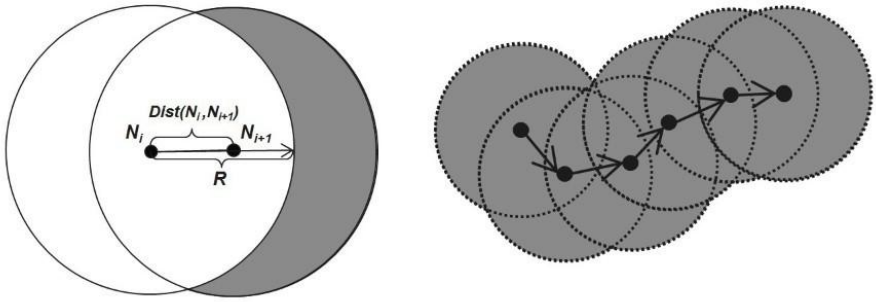
Fu et al. (2010) presented an approach to estimate network density by linear regression technique. Via a routing protocol, such as GPSR, information packet is forwarded from the source node to the Home Node. By collecting information on the route, the network density can be estimated. Fig. 5 shows how this information is updated during the traversal. Assume that the message is forwarded from N_i to N_{i+1}. We denote the number of nodes in the grey area which is newly explored as inc_{i+1}. By keeping updating this variable, Home Node can estimate the network density at the end of Home Node searching phase.

This approach can be illustrated in Fig. 5. A message is transmitted from *Ni* to N_{i+1} and the grey area denoted as E_{Ai} is the newly explored area. Assuming that in_{ci+1} is the total number in E_{Ai}, by keeping adding in_{ci+i} to a variable *Num*, the total number of nodes by far can be calculated. E_{Ai} is formulated as $EA_i = \pi r^2 - H(2r - dist(N_i, N_{i+1}))$, where $dist(N_i, N_{i+1})$ is the distance between N_i and N_{i+1} and r is the transmission range of nodes. As E_{Ai} is negatively correlated with $dist(N_i, N_{i+1})$, the linear technique can be applied to formulate H (*) as follows:

$$H\big(dist(N_i, N_{i+1})\big) = c_1 + c_2 \times dist(N_i, N_{i+1})$$

Where c_1 and c_2 can be calculated based on the linear technique theory in Bretscher (1997).

The total coverage area covered by hop N_i is calculated as $A_i = EA_i + A_{i-1}\, N_{i+1}$. When the message reaches Home Node, total number of nodes *Num* and total area covered by nodes along the route A are generated, thus the network density is estimated by formula $D = \frac{Num}{A}$. As a result, the radius of KNN boundary is formulated as $r = \sqrt{\frac{k}{\pi D}}$.



(a) (b)

Fig. 5. Network Density Estimation

Jayaraman et al. (2010) proposed another approach to estimate KNN boundary in 3D sensor network. Then the expectation of *R* to ensure that there are *k* nodes within the circle with radius *R* can be estimated based on the density. For a given network density ND, the radius of search area R can be estimated as following:

$$Node\ Density\ N_D\ (per\ m^2) = \frac{No\ of\ nodes\ n}{Area\ of\ the\ Terrain}$$

$$Radius\ R = \sqrt[3]{\frac{Area\ A*3}{4*\pi}}$$

However, information of the entire network (number of nodes n and the region Area) is required at the Home Node as the prerequisite of this approach before the estimation, which incurs extra energy overhead.

2.3. Query Dissemination

2.3.1. Explosion Approach

Komai et al. (2011) presented a naive way for KNN query dissemination called Explosion (EXP) method. This partial flooding algorithm focuses on reducing message traffic and keeping high accuracy.

In this method, query node first forwards the message to the Home Node via Geo-routing algorithm proposed in Sanchez et al. (2009). Then the Home Node calculates the radius of partial flooding area (denoted as *r*) and then attaches the *r* to query message. A message flooding is initiated by Home Node. Each node receiving the message first checks the distance between itself and the Home Node. If the distance is smaller than *R*, then it forwards a reply message including the information in terms of itself to the Home Node. If the distance between the node and the query point is larger than *R*, which means it is out of the flooding boundary, it should ignore the message and cancel the subsequent forwarding. In the following section of the thesis, the experimental result shows that this approach generates a high level of accuracy.

However, due to the limitation of the flooding method, collision and traffic congestion are inevitable. Also, Xu et al. (2006) clarified that flooding is not an efficient and energy-saving forwarding method, particularly when nodes are densely packed with much overlapping communication and sensing ranges.

2.3.2. Tree Infrastructure based Approach

Winter and Lee (2004) presented KNN Perimeter Tree (KPT) algorithm to search the locations and IDs of the rest of k-1 nearest neighbour. The basic idea of this infrastructure based approach is to divide the region in the boundary circle into small sub-regions and construct minimum spanning trees in the direction away from the destination.

Fig. 6 illustrates how the spanning trees are established. The nodes first forward the query message to Home Node via GPSR algorithm (Karp and Kung (2000). Once GPSR algorithm enters the perimeter mode, all nodes traversed by this mode are marked as Perimeter Node. We assume N_i is the current perimeter node and N_{i+1} and N_{i-1} are its predecessor and successor, respectively. Each perimeter node builds its own minimum spanning tree which is bounded by three boundaries: two lines from the query point across the midpoint of N_iN_{i+1} and N_iN_{i-1}, respectively, and the arc cut from the search boundary circle by the two lines.

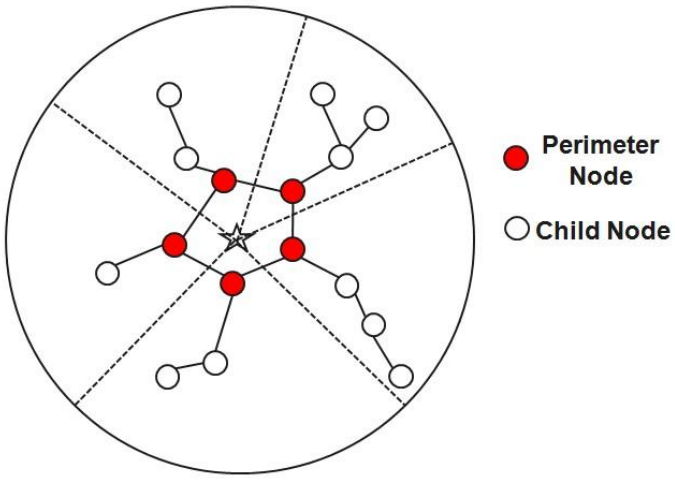


Fig. 6. KNN Perimeter Tree

After boundaries being determined, each Perimeter Node, as the root nodes, begins to establish its spanning tree and then gathers all the information from its children nodes. The Home Node eventually receives all the information about the nodes in the search boundary and sorts them by distance.

Winter et al. (2005) presented another tree structure-based algorithm. This technique is called Single Root KNN Boundary Tree (SRBT) since the tree is rooted at the Home Node. The query message floods to all the nodes within the search boundary. Nodes receiving the message select their parent nodes depending on its approximately geographical position. A parameter called *tree height* denoted as the estimated height of the tree is set before processing the query. Also, a parameter called *level* which indicates the level of each node is generated when nodes receives the

query message. The nodes then set a timer T based on the level of itself and the height of entire tree using the following formula:

$$T = 2 \times Max\{1, TreeHeight - level\} \times MessageDelay$$

Where *MessageDelay* is denoted as the estimated propagation delay between two neighbour nodes. After each node's timer expires, it should report the data of itself to its parent immediately and eventually the Home Node will receive the message from all the children.

Xu et al. (2007) presented a tree based infrastructure algorithm. This approach which is called Geo-Routing Tree Algorithm (GRT) executes KNN query with the algorithm proposed by Goldin et al. (2005). In this algorithm, a parameter named as "*Minimum Bounding Rectangle*" is generated in each node *X* to maintain the smallest rectangle for every child nodes *Y*. The rectangle for *Y* encloses all the locations of nodes rooted at *Y*. For a given query point *q* and a node *u*, a couple of data structures are presented as follows:

MINDIST (u, q): Minimum distance from *q* to *u*'s MBR.

NTable (u): Maintain neighbour information of *u*.

KTable (u, q): Maintain k nearest nodes to *q*.

$\boldsymbol{D_{max}}$: Maximum distance between the query point *q* to sensor nodes from the current *k* closest sensor nodes.

With the data structures above, once a root node receives the query message, it calculates its *MINDIST (0, q)* and generates its *KTable (0, q)*. Then it denotes *Dmax* as $dist_0^k(q, id)$ which is the distance of last entry in *KTable (0, q)*. The node floods the message and any child node *u* receiving the message compares its *MINDIST (u, q)* and D_{max}. If *MINDIST (u, q)* is larger, node *u* then cancels the message forwarding. Otherwise, it sets *KTable (u, q)* based on its *NTable* and *Dmax = Min* $\{Dmax, dist_u^k(q, id)\}$. Eventually, the node forwards the message with new D_{max}. The broadcast continues until leaf nodes are reached, and then the leaf nodes return their *KTable*. The parent nodes in each level aggregate all *KTables* and generate new *KTable* based on the *dist (u, id)*. The aggregation process continues until the message reaches the root node. According to the *KTable* of root node, *k* nearest nodes are generated.

However, GRT algorithm requires tremendous energy consumption to construct the infrastructure if the network topology changes. Also, the large amount of data structures may lead to enormous storage space and message overhead.

2.3.3. Itinerary based Approach

Itinerary based KNN query processing, which propagates queries and collects data along a predetermined itinerary, is an infrastructure-free technique (Wu et al. 2007). In this approach, we denote two kinds of nodes, Q-node and D-node. As to the first Q-node, the Home Node initiates query process which follows these steps: gather the data (e.g. distance to the query point, sensing data) from its neighbours (D-node), decide next Q-node by greedy method (the farthest distance from the current Q-node along the predetermined itinerary direction to its neighbours) and forward the data to itself. This progress is repeated by all the Q-nodes until the query reaches the search bound. Then the last Q-node forwards the message directly to the Home Node. Fig. 7 illustrates the basic idea of these two kinds of nodes. In this example, message delivery is generated by A and along the itinerary. Initially, node A sends the probe message to its neighbours (B, C, D and E) to gather their data. After receiving the response of all its neighbours, node A then chooses E as next Q-node due to the fact that node E achieves largest progress along the itinerary direction among A's neighbours. Then node A inserts the data about its neighbours and itself in the query message and forwards it to E. E repeats the algorithm again and forwards the message to next Q-node. This algorithm keeps operating repeatedly until the KNN boundary is reached and the query message will be forwarded with the data collected so far back to home robot for processing. By this means, all the

robots within the KNN boundary is expected to be visited and their data is obtained by the Home Node.

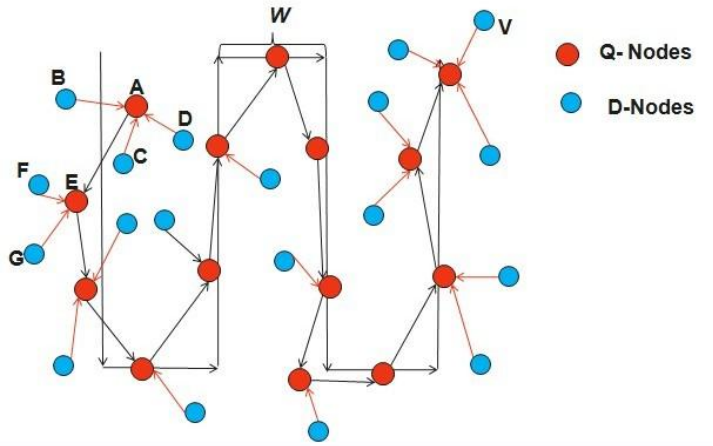
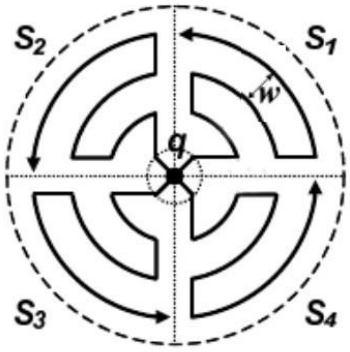


Fig. 7 Nodes in Itinerary Based Approach

Wu et al. (2007) proposed DIKNN algorithm. This algorithm first determines the itinerary width *w*, as shown in Fig. 7. Apparently, a small *w* leads to a denser itinerary traversal but generates more power consumption, while a large *w* results in fast message transmission but low accuracy. In this approach, $w = \sqrt{3}r/2$ is selected as a good balance of accuracy and efficiency. The rationality of this selection was also proved by Xu et al. (2006). As illustrated in Fig. 8, this approach divides the KNN boundary into multiple sections. Query dissemination is processed individually in each section.



.

Fig. 8 DIKNN Query Dissemination

When a query Q is initiated by the Home Node, the query message is disseminated within the search area in parallel. As presented at the beginning of this section, after message is traversed with the routing algorithm, it should be forwarded back to the Home Node. Then the Home Node aggregates all the information about the nodes within this area and *k* nearest nodes can be generated by sorting them by distance. To avoid collision and traffic congestion, a parameter timer is generated whenever a D-node receives a data inquiry message from Q-node. In DIKNN timer is denoted as $timer = (\frac{\alpha}{2\pi})im$ where α is the angle formed by the itinerary line, the line between current Q-node and D-node, and *i* is the receiving precedence of the message, and *m* is a time unit for Q-node to wait for D-node to respond for its data query message. The D-node should not respond to Q-node until the timer expires. A method to improve fault tolerance in DIKNN is presented by Wu et al. (2008). In this method, a D-node receiving probe message knows the relative reply

precedent with its neighbour D-nodes. It caches the response with high precedence. If no acknowledgment is received by Q-node, this D-node should send cached data along with its own data to Q-node. This method enables the algorithm to overcome package loss caused by D-nodes.

DIKNN integrates query propagation with data aggregation and performs well in terms of KNN query processing. However, a research conducted by Kim et al. (2013) also claimed that DIKNN algorithm results in excessive energy consumption and high query latency because of the lack of method to optimize the KNN boundary setup progress.

Fu et al. (2010) presented an itinerary approach called Parallelizing Itinerary-Based KNN Query Processing (PIKNN). The itinerary topology in PIKNN has been illustrated in Fig. 9. Unlike other itinerary-based KNN processing algorithms, the parallel concentric-circle itineraries for PIKNN enable more KNN query threads than the number of itinerary generated during the progress of propagation.

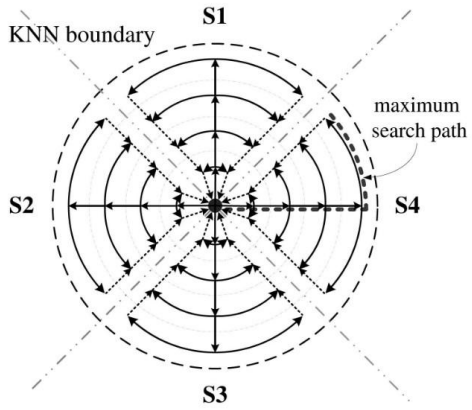
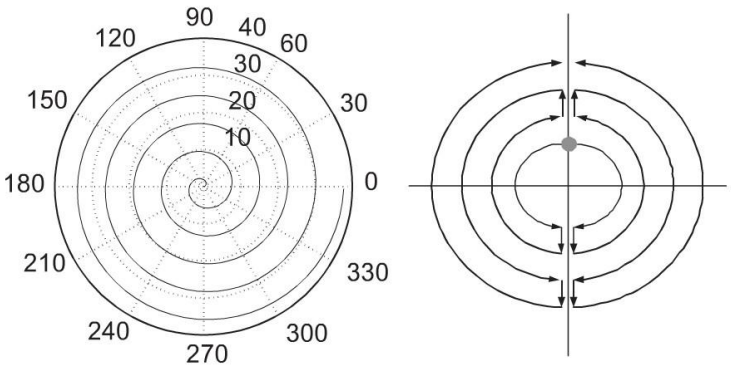


Fig. 9 Parallel Concentric Itineraries in PCIKNN

In this thesis, an innovative idea is proposed to update KNN search boundary dynamically. The authors first define two different schemes: minimum energy consumption mode and minimum latency mode. In the minimum energy consumption mode, when a KNN query reaches a sensor node at the KNN boundary, the KNN query will wait at the boundary node for the control message from the Home Node. The Home Node should check the received message periodically; if the number of candidate nodes is smaller than *k*, a control message to expand the search boundary should be initiated at the Home Node; if the nearest *k* node has been collected, Home Node will send a message to end the aggregation progress. In minimum latency mode, the goal is to minimize the total query latency. Thus, this scheme allows nodes to forward the query message even when it is out of the search boundary. Specifically, when the query message reaches a boundary node, this node will hold the message for a period of time to wait for the control message from Home Node. If no successor message is received after the timer expires, the node will propagate the query message to the region out of the boundary to collect more data. If the node receives the control message from Home Node which indicates that the result of KNN has been found, the propagation should be terminated.

Xu et al. (2007) presented IKNN algorithm. The authors proposed two formats of IKNN in this work. In the first approach, the authors implemented itinerary-based KNN query processing with Archimedean

spiral, as shown in Fig. 10 (a). The Archimedean spiral obtains a constant separation distance, which equals to $2\pi b$. This feature enables us to control the density of the itinerary. The query message dissemination is initiated by the Home Node. Then the message is transferred along with the predetermined Archimedean spiral via greedy algorithm and aggregates the information about the traversed nodes. Once the list of *k* nearest neighbour is obtained, this message will route back to the Home Node. The second approach is illustrated in Fig. 10 (b). Different from the first approach, two parallel query messages are generated at the Home Node and propagate respectively along with two predetermined itineraries. Two itineraries have to meet periodically to check the search result and once the total number of nodes stored in two query messages reaches *k*, the propagation is terminated.



(a) Archimedean Spiral (b) Parallel Itinerary

Fig. 10 IKNN Algorithm

2.3.4. Geocasting Based on Depth-First Search

Existing solutions make a tremendous contribution to solve KNN query processing problem. However, to the best of our knowledge, none of these existing approaches guarantees the delivery to all the robots within KNN boundary. For instance, these algorithms may ignore some special nodes which are geographically near to the query point but whose predecessor nodes are out of the search region. The example shown in Fig. 11 indicates that node A is within the search region and obtains a smaller distance to query point than node B. However, due to the fact that node A's predecessor node (denoted as node C) is out of boundary, search algorithm determines to ignore all the successors of node C and thus node A is ignored by mistake. The same situation occurs with robot E and robot G.

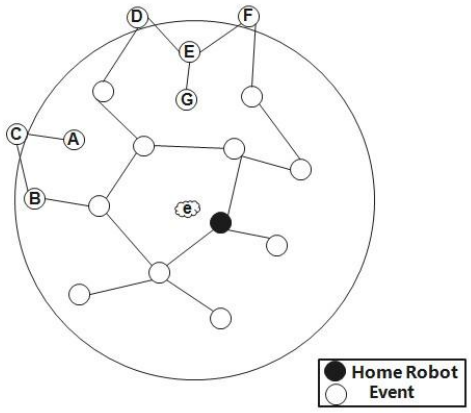


Fig. 11 Non-Guaranteed Delivery

To overcome this issue, Bose et al. (2001) presented a mono thread geocasting algorithm called *Geocasting Based on Depth-First Search*

Traversal of The Face Tree (GBDFS) to achieve the guaranteed delivery to all the nodes connected to the source node. Multipath algorithm can be applied to improve the performance. This algorithm maintains a very small amount of data stored in nodes and significantly lowers the overhead of the message. Stojmenovic et al. (2004) extended this algorithm to implement guaranteed delivery with geocasting. This approach ensures that every node encircled by KNN boundary is traversed. To achieve this algorithm, some concepts are proposed and illustrated.

Artificial Node: Artificial node is a virtual node generated by the algorithm. This node does not really exist and can be only for reference use. To generate an artificial node, we should first choose a face and select a random point within this face as the artificial node of this face.

Entry Edge: Entry edge is an entry for one face to another unvisited face. In other words, entry edge is the connection between child face and its parent face. Several keys are proposed to compare all edges of a face and select the entry edge. Given a face *f* and an artificial node *S* in this face, the primary key to determine entry edge *entry (f, S)* is the distance of the edges of *f* to *S*. This distance is determined by the distance from *S* to the nearest point of the edge. In some cases, two or more edges share one closer point (denoted as *C*). The second key is presented to compare

these edges. For two points *a* and *b*, let $\overrightarrow{ab}$ be the direction from *a* to *b* measured in radian, counter clock wise from the positive x-coordinate, the entry edge is determined by $\overrightarrow{SC}$. The third key is the angle formed by *S*, the nearest point from the edge to *S* and the other end point of *S* (denoted as *D*). If these keys are not enough to select the entry edge, the last key is $\overrightarrow{CD}$ Bose et al. (2001) proved that these four keys are sufficient to identify two unique edges.

Entry Edge is a door between the parent and the child during the face tree traversal and is generated dynamically. Every time an entry edge encountered, the face tree obtains a new child during this progress.

The first phase is the face tree expansion and traversal progress. In this algorithm, the query point (denoted as *S*) is the artificial node [which was described in Stojmenovic (2004) and Morin (2001)] during one single progress of KNN query processing. A circle area *A* with centre point *S* and radius *r* is generated to ensure that the propagation is restricted within this area. The value of *r* is determined by the KNN Boundary Estimation Algorithm.

Initially, a query message is received by the Home Node, Home Node estimates KNN boundary and generates a new query message with additional data (e.g. the coordinate of the query point, the KNN boundary *R* etc.) to travel with the progress of the face tree traversal rooted at the Home Node. Whenever the message reaches the edge currently being traversed (denoted as *e*), it checks whether *e* is the entry edge of *f* or its

opposite face. If *e* is its opposite face's entry edge, then the message stops traversing the edges of *f* and enters the newly explored face. Afterward, a new traversal progress is initiated in the new face (denoted as *f'*). As the child of *f*, *f'* starts the traversal in the set of its edges. This face tree expansion progress continues until the message reaches the face with no new entry edge (denoted as leaf face). In this case, edge traversal will reach the entry edge of current face eventually. Thus the message will be returned to its parent and the parent continues to implement this algorithm until new entry edge is explored or the traversal reaches its own entry edge. In order to restrict the search within KNN boundary, the message does not traverse in the face with all edges out of KNN boundary. Ultimately, the message reaches the Home Node with all the data aggregated from the robots encircled by KNN boundary. For a given geocasting area, Morin (2001) proved that every robot within this boundary will be traversed via this algorithm.

An example of GBDFS is presented in Fig. 12. Assume that the event occurs in f_1 and thus denoted as artificial node *S*. CB is the first edge of f_1 to be traversed and it is also the entry edge of f_3, thus the query message enters f_3 and continues to traverse the edges in f_3. EB is explored as the entry edge of f_4, thus the message visits f_4. Please note that EF, JF and JE are all out of KNN boundary and as a consequence, we will ignore the face which is consisted by them even though EF is the entry edge. There is no other entry edge in f_4 and the traversal in f_4 eventually goes back to

EB which is the entry edge of f_4. Then the message returns f_3 which is the parent of f_4 to explore more potential entry edges. This face tree traversal progress remains active until the traversal in f_1 reaches edge BC again which indicates that the traversal has completed and all the robots within KNN boundary is traversed.

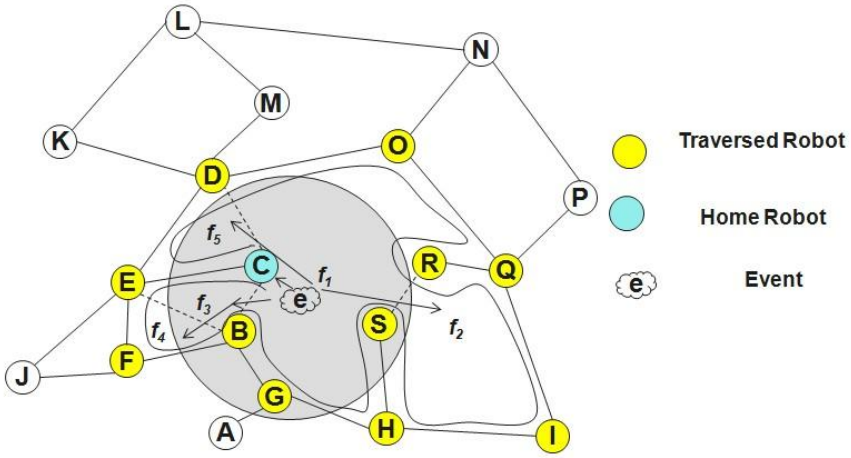


Fig. 12 GBDFS for KNN Query Processing in WSRNs

Chapter 3: KNN Query Processing in WSRNs

3.1. Home Robot Search in WSRNs

The goal of the first phase of KNN query processing in WSRNs is to find the Home Robot (corresponding to the Home Node in KNN query processing in WSRNs). We assume that an event occurs at a random point at the sensor layer. This event is sensed by one of the sensors in the sensor layer and then forwarded to a vicinity robot (may not be the closest one to the query point). This robot then generates a query message, including the information of the event and the desired number of target robots, and forwards it to the nearest robot to the query point.

Recall from Section 2.2, GPSR and BLR can be both implemented to search for Home Robot after the query message is received by a robot in the vicinity (denoted as Vicinity Robot) of the query point. However, Akyildiz et al. (2005) clarified that BLR routing is not a guaranteed delivery and Sanchez et al. (2009) verified that BLR generates an enormous number of duplicate message which significantly lowered the performance of the algorithm. For these concerns, we choose GPSR as our routing algorithm to search for Home Robot.

The Vicinity Robot inserts the information of the event into the query message and starts a GPSR routing with the information about a destination point of the query point. Winter and Lee (2004) verified that

the query message will eventually reach the nearest robots around query robot, which can be confirmed by routing in Perimeter Mode around the query point.

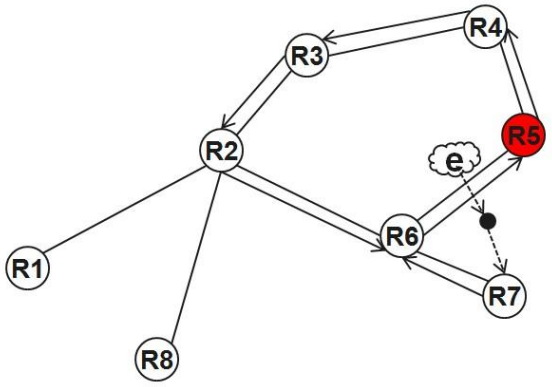


Fig. 13 Home Robot Search

Fig. 13 illustrates an example of Home Robot Search in WSRNs. An event is sensed by a sensor in the sensor layer and this sensor forwards the message to R7 which is in the vicinity of it. Then R7 generates the query message and initiates the routing to search for Home Robot via GPSR. The Greedy Mode is terminated and the routing switches to Perimeter Mode when R5 receives the query message. After the query message traveling around the query point, the Home Robot can be determined. Afterwards, the Home Robot should terminate the routing and prepare to initiate the next phase.

3.2. Boundary Estimation in WSRNs

As we discussed in Section 2.3.1, propagating the query message to the whole network to search for *k* nearest robot to interact with certain event is inefficient and will result in tremendous latency. As a result, a search boundary should be estimated to restrict the propagation within a certain boundary. Max Hop Distance (MHD) (Winter and Lee, 2004) indicates the partial network density. However, estimation based on this algorithm could be less accurate when the distance between two certain neighbours is extremely large. Then the boundary will be overestimated and unnecessary energy consumption occurs. Other methods based on network density generate enormous number of messages while collecting the data for estimation (Fu et al. 2010) or requires the network density data before query processing (Jayaraman et al. 2010). In most cases, the density of the network is unknown and varies from region to region, thus partial network density around Home Robot should be considered for estimating the KNN boundary. To increase the accuracy of the estimation and reduce unnecessary communication between robots, we propose a method to estimate the boundary in WSRNs network based on partial network density around the Home Robot.

The first step of our algorithm is to aggregate the hop distance information around Home Robot. As we mentioned before, in our distributed system, every robot knows the information about its neighbours and itself, including their coordinate. Thus, the distance

between every robot and its neighbours can be calculated and stored in the memory. After the initialization of the Home Robot, a query message is generated and sent to every neighbours of the Home Robot. This message will only be forwarded to Home Robot's neighbours and should not be propagated further. Each robot receives the query message and then calculates the average distance between its neighbours and itself and then adds this parameter K_i to the query message where *i* indicates that this robot is the *ith* neighbour of the Home Robot. Eventually, the query message should be sent back to its source (i.e. The Home Robot). Denoting the average distance between the Home Robot and its neighbours as *k* and the degree of the Home Robot as *M*, the Home Robot estimates average hope distance after receiving *i* messages from its neighbours. The partial average hop distance *d* is given by:

$$d = \frac{K + \sum_{i=1}^{M}(K_i)}{M + 1}$$

As discussed in Section 2.3, an overestimated KNN boundary results in large energy consumption and latency thus diminishing the performance of the algorithm. On the contrary, an underestimated KNN boundary may lead to the failure of the first search (i.e. the number of robots within the search region is less than *k*) and enforce the Home Robot to initiate a second search with an expanded KNN boundary which also is inefficient. Thus, the basic challenge of KNN boundary estimation is to optimize the boundary radius. In KNN query processing, we assume that the target number of nearest neighbour we are searching for around

query point is *k* and the radius of the search boundary is *r*. The ideal situation is that there are exactly *k* robots within the KNN boundary centred by query point with a radius *r*. With the purpose of increasing the possibility of the occurrence of this event, we proposed a network estimation algorithm. Before that, some discussions are provided below to clarify the relationship between *r, k* and the density of the network *D*.

We assume that all robots are deployed randomly. A circular area centred by a random point (marked as query point in our example) with radius *r* is generated as the target area (denoted as *A*). In view of the fact that the deployment of each robot is an independent event, the possibility that a robot is encircled by this area is equal to the ratio of the target area to the total area. Assuming that the total number of deployed robots is *N*, denoting network density *D* as *N/A*, then the expected number of robots *n* within this area is given by:

$$n = \frac{N\pi r^2}{A} = D\pi r^2$$

Where *A* is denoted as the total area. Thus, *r* should be a positive correlation with *k* to enable *n* to approach *k*. As presented in Equation, for a fixed *A*, *r* is positively correlated with n. For a fixed *n*, *r* is negatively correlated with *D*.

As a result, *D* and *k* are essential parameters to estimate KNN boundary. The average hop distance of robots neighbouring Home Robot, to some extent, indicates the partial network density around query point. It can be verified that a looser network should result in a larger *d*. For

these concerns, after d is obtained, we proposed a linear equation to approximately estimate the radius of KNN boundary r:

$$r = kdc$$

Where c is adjustable coefficient. The most essential part of this equation is the value of c and we will optimize the value in the experiment later. The calculated r is kept in the query message and is prepared to be forwarded in the next phase.

An example of our boundary estimation can be illustrated in Fig. 14. As we can see, an event marked with star occurs randomly in the network and a sensor forwards the message to the nearest robot A (the Home Robot). Then the KNN boundary estimation phase is initiated by the Home Robot and propagated in robot network. Home robot then sends a query message to all its neighbours (B, C, and D). Then B, C and D calculate the average distance between themselves and their neighbours d_B, d_C and d_D, respectively, and send these parameter back to robot A. Robot A will wait until 3 messages received (the degree of A is 3) and then calculate r with Equation 1.

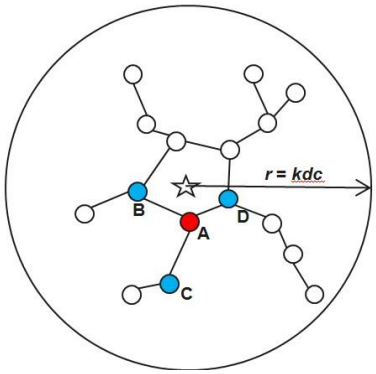


Fig. 14 KNN Boundary Estimation

3.3. Query Dissemination in WSRNs

The most essential and complex part of KNN query processing is to disseminate the query message to robots encircled by KNN boundary. Multiple solutions have been presented in Chapter 2. However, to our best knowledge, there are no satisfactory localized solutions to KNN query dissemination in terms of energy consumption, message overhead and latency. Thus, we present three possible solutions to improve the performance of this progress.

3.3.1. Multiple Auction Aggregation

3.3.1.1. Introduction

Recently, some research has been conducted to implement Auction protocol in WSRNs. Melodia et al. presented a localized single-task, single robot auction protocol in Komai et al. (2011) and a survey article (Dias et al. 2006) summarized some centralized or localized auction protocols applied in WSRNs. Mezei et al. (2010) proposed a novel auction protocol named Auction Aggregation Protocol. In this algorithm, query message is disseminated by tree expansion and tree construction. Tree expansion starts at the Home Robot by creating a tree rooted at collecting robot. Query message is forwarded to every robot neighbouring it, and each robot receiving the message is aware of its parent robot through the identification number. Each node can only choose one robot as its parent robot and thus becomes leaves if they do

have any other child robot. Bidding information is kept in the query message to record the best bidding so far. Specifically, each node receiving the message will compare the bidding stored in the message with its own bidding and replace it if it offers better bidding. Eventually, the best bidding so far is stored when the message reaches leaf robot. In order to prevent the message from flooding to the whole network, each robot should relay only when it is within a distance of k hops from collecting robot. If the robot is k hops away, the message will be dropped.

Inspired by this solution, we present a new algorithm to aggregate multiple bidding within a KNN Boundary. We use the same idea to implement the tree expansion. In order to ensure that each robot only has one parent, we denote an identification flag *ischildrobot* whose default value is *false*. Each robot should set the value of this flag as *true* to clarify that it has been a child robot and drop sequential messages. More schemes about this algorithm will be verified in Chapter 4.

As a distributed system, robots will only know the information about its neighbours and itself. Hence, in order to better implement our algorithm, some basic data structures should be generated after initiating the robot. These structures are presented as follows:

d (u): Maintain average distance between u and its neighbours.

C(u): Maintain the coordinate of robot u.

NTable (u, q_i): Maintain the information about its neighbour q_i.

Once the boundary estimation (implemented via the method in Section 3.1) is completed, a query message is generated to implement KNN query dissemination. Some essential data structures should be constructed by the Home Robot. These structures are proposed as follows:

r: maintain the radius of KNN boundary

KTable : Maintain the list of *k* Nearest Robot by far (initiating by null).

m(u): The current size of ***KTable*** until *u*.

C_{event} ***(u)***: Maintain the coordinate of query point.

With these parameters, our Multiple Auction Aggregation algorithm (MAA) is described as follows. MAA algorithm consists of 2 phases: message propagation and tree expansion phase and backtracking phase. In message propagation and tree expansion phase, the Home Robot forwards the query messages to the robots within the KNN boundary to gather the information along with the routing and construct the tree rooted at Home Node. This phase continues until a leaf robot or a robot out of KNN boundary is reached and then turns to backtracking phase. The backtracking phase aims at aggregating the data in the messages and forwarding it back to the Home Robot to process. In the next section, we will describe these two phases in detail.

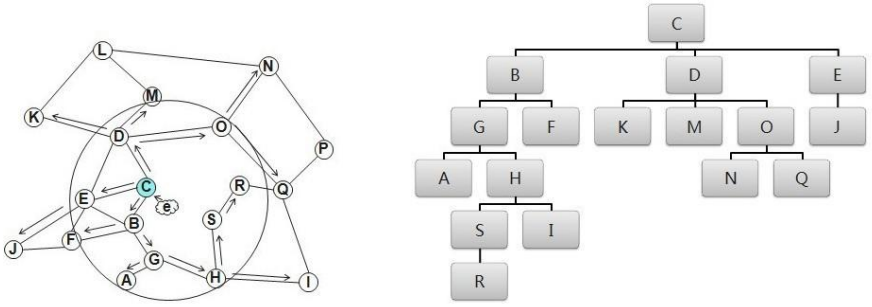
3.3.1.2. Message Propagation Phase

After the boundary estimation, the Home Robot forwards the message to all its neighbours. Each robot first checks whether it has been other robots' child and drops the message if so. Afterwards, this robot inserts itself into the list *KTable* if $m(u) < k$, which indicates that the *KTable* is not full yet**.** Otherwise, this robot calculates the distance between query point and itself by $C(u)$ and $C_{event}(u)$ and compares the value with the values in *KTable*. If the value is smaller than the value of any robot in the list (i.e. offers a better bidding), it will add itself to the table and remove the last object in the list. Eventually, Robot *u* forwards the message to its neighbours.

The message is forwarded to the network with the expansion of robot tree until a leaf robot or a robot out of KNN boundary (this can be verified by comparing the distance between the robot and the query point to *r*) is reached. Then the back transmission mode is activated to forward a backtracking message with information gathered along with the routing back to Home Node for processing. To avoid collision and traffic congestion, we keep the branch information in the message and every message should route back exactly based on the trace.

An example of message forwarding and tree expansion in MAA is illustrated in Fig. 15. For simplification, we assume in this case $k = 1$, $u = 1$, which means every query message will only keep one best bidding. In Fig.15 (a), an event occurs at a random point in the network and is

sensed by a node. The node forwards the query message to the nearest robot C, which is denoted as Home Robot. Then robot C initiates the Boundary Estimation phase and generates the radius of KNN boundary *r* and all the sequential processing should be restricted within this area. Afterwards, query dissemination phase is initiated by forwarding the query message to all the neighbour robots of C (i.e. D, E, and B). Meanwhile, the tree rooted at robot C is generated. Robot D, E and B mark them as the 1st level in this tree structure and insert this information with its bidding (i.e. distance to the query point) into the query messages, respectively. Afterwards, the query message is forwarded to the next level in the tree structure (i.e. G, F, K, M, O and J). It should be noted that robot F and J will stop succeeding forwarding and enter backtracking phase to create the backtracking message including the bidding information received so far and forward it to their parents after data processing as they are out of the search boundary. The tree expansion and data aggregation progress continues until the robots out of KNN boundary or a leaf robot are reached. The result of tree expansion is presented in Fig. 15 (b).



(a) **(b)**

Fig. 15 MAA Tree Expansion

3.3.1.3. Backtracking Phase

At the end of the previous phase, the query messages are kept by the leaf robots. The next challenge of MAA is how to implement data aggregation after the tree expansion is completed. As the tree structure we presented, a parent robot always generates multiple children and thus should receive multiple backtracking messages. To aggregate the completed information about the child robots, a parent node should wait for the backtracking messages from all its children before creating its own backtracking message and forwarding the message to its own parent. A fault tolerant scheme can be added in this progress to decrease the impact of the loss of backtracking messages by children on the algorithm. Specifically, each robot will generate a timer after receiving the first backtracking message from its child. After this timer expires, this robot should process the data and generate its backtracking message despite the data from its non-response children, and then forward the message to its

parent. This scheme helps MAA to overcome the package loss from children but may miss the information from the children with a long branch (which means the response time of this child may be much longer than other children and even longer than the timer). Thus optimization should be conducted to balance this probability and network efficiency.

After receiving the entire backtracking messages from children or the timer expiring, robot will process the data and generate its own backtracking message. A naive approach is to simply integrate all the *KTable* from its children and create a new backtracking message. However, at the last few hops in backtracking mode, this approach may generate an enormous backtracking message and the size of the message may reach an unacceptable level. This situation becomes even worse in a dense network. To enhance the performance of MAA, instead of simply integrating all the bidding from children, the bidding information from children is compared by their parent and only k best bidding is selected and added to new backtracking message (In our case, $k = 1$). Hence, this algorithm restricts the size of the backtracking message and thus significantly reduces the latency.

Assume that Home Robot generates m children ($m >= 1$). Ultimately, the Home Robot receives mk biddings, sorts them by distance to the query point and selects the k best among them. We also consider the situation that less than k robots are collected due to underestimated KNN boundary. In this case, Home Robot should restart MAA with a larger r.

Generally speaking, this incremental situation is much more inefficient than the one with overestimated boundary due to the energy and time wasted on the second search. We simply double the KNN boundary after the first failure to lower the possibility of the second failure.

3.3.2. Partial Depth First Search

3.3.2.1. Introduction

Depth First Search is a classical algorithm extensively used to traverse all objects in a particular graph. This algorithm is initiated by one root and visits as far as possible before the message backtracking. Some studies have been conducted in terms of solving KNN problem with Depth First Search. Samet (2003) proposed a depth first search based algorithm to implement *k* nearest neighbour search. In this approach, depth first method is used to estimate a minimum boundary within which a nearest neighbour can be found. The author also clarified the maximum distance at which the nearest neighbour is possibly found. Yu et al. (2006) proposed a KNN search algorithm based on depth first search. This algorithm uses depth first search to determine the minimum radius and the effective radius of a hyper ball and then select all the nodes in this hyper ball as the candidate *k* nearest neighbour of the root.

Motivated by these relative works and the features of depth first search, we propose a depth first algorithm to visit the robots within KNN boundary and aggregate the data of these robots to implement KNN search in WSRNs. The basic idea of how we implement depth first search is by generating a first-in last-out stack in the query message to record the robots that have been visited by far and their sequence. An example

to illustrate the mechanism of this scheme is presented below. The key of the visiting sequence is *ID*.

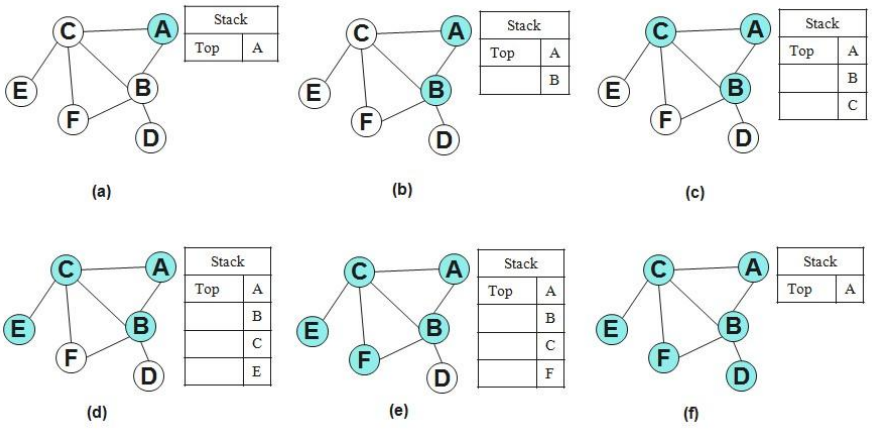


Fig. 16 Depth First Search

In Fig. 16 (a), a stack is generated. A is pushed onto stack. A has two unvisited nodes, among which B obtains the lowest ID. Choose B as the next target. In Fig. 16 (b), B is pushed onto stack. Choose C as the next target. In Fig. 16 (c), C is pushed onto stack. Choose E as next target. In Fig. 16 (d), E is pushed onto stack. E has no unvisited neighbour. E pops out from stack. Backtrack to C. C has unvisited neighbour F. Choose F as next target. In Fig. 16 (e), F is pushed onto stack. F has no unvisited neighbour. F pops out from stack. C is currently at the bottom of stack. Backtrack to C. C has no unvisited neighbour. C pops out from stack. B is currently at the bottom of stack. Backtrack to B. Choose D as next target. In Fig. 16 (f), D is pushed onto stack. D has no unvisited neighbour. B is currently at the bottom of stack. D pops out from stack.

Backtrack to B. B has no unvisited neighbour. B pops out from stack. A is currently at the bottom of stack. Backtrack to A. A has no unvisited neighbour. A pops out from stack. Eventually, stack is back to null and the depth first search is completed.

3.3.2.2. Message Routing

The basic idea of our routing and data aggregation progress in PDFS is to implement depth first search within KNN boundary to collect data from all the robots and then forward the message to Home Robot for processing. This algorithm is given as following. Assume that r is the robot currently being visited, T is the forwarding table determined by ID, r_{next} is next robot to be visited. $r.distance$ is the distance between r and the query point, R is the radius of KNN boundary, and $2s$ is the stack in the query message for recording the visiting sequence.

Algorithm Depth First search for *k* Nearest Neighbour

```
1:repeat
2:      if r.distance ≥ R then
3:              r pops out from s
4:              r_next = r's parent
5:      else if r is at the bottom of S then
6:              if T==null then // If r has no unvisited neighbour
7:                      r pops out from s
8:                      r_next = r's parent
9:              else if T ≠ null then // If r has unvisited neighbours
10:                             r_next = next robot in T
11:                     end if
12:             else
13:                     r is pushed to s
14:                     r_next = next robot in T
15:             end if
16:     until s == null
```

More specifically, we first denote the Home Robot as the root of the search tree. Then the Home Robot chooses a neighbour of itself as its child and forwards the query message to this robot. The key of choosing child is the *ID* of the neighbour robots (lowest *ID* first). The message should record the visiting sequence by *s*. After receiving the message, *r* first examines whether it is located outside of the KNN boundary (this can be implemented by comparing its coordinate and the coordinate of the query point) and returns the message to its parent robot if so (i.e. mark itself as leaf robot). If *r* is within the KNN boundary, *r* inserts the information of itself to the message and checks if there is any unvisited neighbour existing. If multiple unvisited neighbours exist, *r* chooses one

of them by *T* and forwards the query message to this robot. If all the neighbours of *r* have been visited, the message will be forwarded back to *r*'s parent and operate the algorithm again. Eventually, query message is forwarded back to the Home Robot and all the neighbours of the Home Robot have been visited. The query message maintains all the information of visiting robots which enables the Home Robot to sort them by distance to the query point and choose *k* nearest ones.

A demonstration of PDFS is presented in Fig. 17. The query message is first received by Home Robot C. After C's initialization, C inserts the information of itself and forwards the query message to one of its child. In our example, the sequence of traverse children is determined by key = ID, which means the robot with the lowest ID will be visited first. In this case, B is selected as the next robot to be visited. Then E, D, M and L are visited in sequence. Note that L realizes that it is out of KNN boundary and thus forwards the query message back to M. This progress lasts until the message is forwarded back to C and no unvisited neighbour exists. Ultimately, C extracts the data in the query message and sorts the candidate robots by distance to choose the *k* nearest ones.

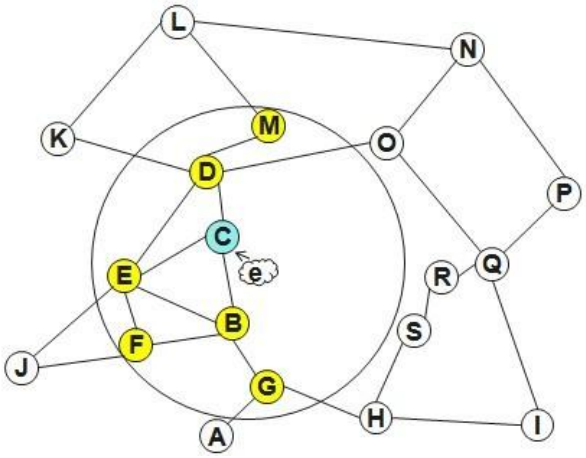


Fig. 17 PDFS Algorithm

PDFS is fairly sensitive to package loss which may result in unrecoverable and catastrophic consequence (i.e. the failure of the entire routing). Thus we present a fault tolerant scheme which can be implemented to PDFS to lower the impact of package loss. Assume that query message *m* arrives at robot *r* for the first time. *r* stores a copy of *m* in its memory before relaying the message. After that, every time *r* receives the new query message from its child with updated information. *r* replaces *m* with the new message. *r* should delete this backup message to save the memory space when it enters the backtracking mode (i.e. all the children of *r* have been visited). In the case that *r* does not receive any response from its child for a predetermined period of time (can be achieved by generating a timer after forwarding the message), *r* sends the backup message in its memory to its next child in the forwarding list. The none-response child is considered as "visited" and any sequential message from this child will be disregarded to avoid duplicate messages

in the network. The information from the non-response child and its successors will be lost but this scheme prevents the entire routing from failure when package lost occurs. An overestimated timer results in large latency when package loss occurs， but an underestimated timer may lead to unnecessary loss of data when the latency of the entire network is fairly large. As a result, the timer should vary with the density of the network and the level of the search tree at which the robot is. In most cases, the robots in a dense network or in vicinity of the root suffer larger latency for waiting for the response from children. The drawback of this scheme is that extra storage space for the backup message in robots is required.

To conclude, PDFS is an accessible algorithm to solve KNN query processing in WSRNs with low protocol complexity. However, this algorithm requires fairly large space to keep the routing stack which enlarges the size of the query message. Also, PDFS is not a guaranteed traversal algorithm.

Chapter4 Simulation

4.1. Comparative Algorithm

To evaluate the performance of our proposed methods, we implement the IKNN proposed by Xu et al. (2007) as the comparative algorithm (this algorithm is also mentioned in Section 2.4.3). We make some minor change to optimize IKNN and add a recovery mode to IKNN. Recall from Section 2.4.3, an itinerary based algorithms (e.g. IKNN). Q-node aggregates the data of its neighbour nodes (i.e. D-node) and forwards the message to next Q-node in the itinerary. These itinerary based algorithms (Fu et al. 2010, Chempavathy and Vijayaraja, 2010, Wu et al. 2007, Winter et al. 2005 and Xu et al. 2007) assume that the network is dense enough and next Q-node can always be found. This assumption is not always reasonable if the network is loose. Xu et al. (2006) verified this problem and denoted this phenomenon as *Itinerary Void*. The main cause of this problem is that in itinerary based algorithm the next Q-node is determined by Greed Protocol, which means the current Q-node chooses the node closer to the itinerary as the next Q-node (makes geographical progress along the itinerary). However, greedy protocol does not guarantee the packet delivery (Akyildiz et al. 2002) and may fail if no node is closer to the itinerary. For ease of understanding, Fig. 18 shows an example of the failure of message delivery in IKNN. The curve stands for itinerary and the red nodes stand for Q-node. Node C fails to find next

Q-node because none of its neighbours can make geographical progress along the itinerary.

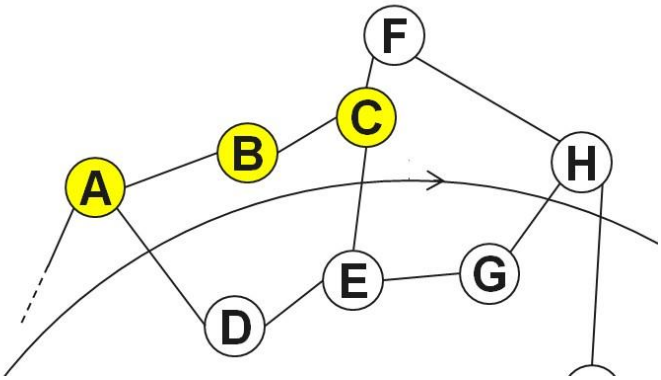


Fig. 18 Failure of Message Delivery in IKNN

To address this drawback, we propose a scheme to improve IKNN. The main idea of this scheme is to create a recovery mode to help message recover from failure and maintain routing after failure occurs. Also, a scheme is presented to standardize the itinerary initiation. The main mechanism of IKNN has been addressed in Winter et al. (2005). Thus, in Section 2.4.3 we mainly focus on these two newly explored schemes discussed in the next section.

4.1.1. Itinerary Initiation Based on Home Robot

In itinerary based algorithms, the first phase is to set up an itinerary and enforce the robot to route over it. In IKNN, the itinerary is an Archimedean spiral centered at query point and initiated at Home Robot. The relative position of the Home Robot and query point is random and

generates a great influence on the routing performance at the start of the routing. An optimal generated itinerary enables the algorithm to avoid failure at the start of the routing and further improves the accuracy. For instance, in Fig. 18, if we set the start point of the itinerary as (w, 0), then Home Robot will enter recovery mode at the beginning as there is no nearer robots with this point. As a result, the initiation of the itinerary should be designed carefully to enhance the performance of the algorithm.

The basic idea of our scheme is to create a coordinate system with an origin of the query point (denoted as *Event Coordinate System*), and initiate the itinerary based on the quadrant of Home Robot in this coordinate system. Assume that the Archimedean spiral is generated in clockwise direction, let c be the coordinate of the Home Robot in *event coordinate system*, i be the start point of the itinerary and w be the width of the itinerary, this algorithm can be illustrated as following.

Algorithm . Itinerary Initiation in IKNN

1:**if** c is at 1st quadrant **then**
2: $i = (w,0)$
3: **if** c is at 2nd quadrant **then**
4: $i = (0,w)$
5:**if** c is at 3rd quadrant **then**
6: $i = (-w,0)$
7:**if** c is at 4th quadrant **then**
8: $i = (0,-w)$
9: **end if**

An example is proposed in Fig. 19 to demonstrate this algorithm. In this example, the home C is located at the second quadrant in the *event coordinate system*. Based on the algorithm we proposed, the start point of the itinerary should be (*0, w*). The main purpose of this algorithm is to make the start point of the itinerary as near as possible (in clockwise direction) with the Home Robot to lower the probability of failure at the initiation of the itinerary.

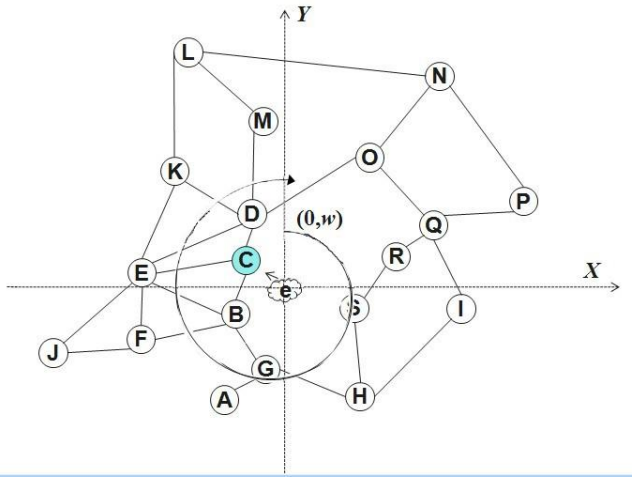


Fig. 19 Itinerary Initiation Based on Home Robot

4.1.2. Recover Mode of IKNN

Recall from Section 4.1, Q-node may fail to find next Q-node if there is no nearer nodes to itinerary. To ensure the operation of the algorithm, we present an algorithm to recover IKNN from failure.

The basic idea of this algorithm is to forward the query message to one random neighbour of current Q-node if the basic mode of IKNN fails.

Afterwards, this randomly chosen node checks whether it obtains a nearer neighbour to the itinerary and chooses it as next Q-node if so. In the case that there is still no nearer neighbour to itinerary, this node will enter recovery mode again and choose one random neighbour to forward the query message. Recovery mode continues until one of the nodes is eligible to enter the basic mode of IKNN (i.e. obtains a nearer neighbour to the itinerary). The recovery mode will only be applied when the basic mode of IKNN fails.

Note that recovery mode may generate an endless loop if the failed Q-node chooses its previous Q-node as the target of recovery mode. In Fig. 18, node C enters recovery mode due to the fact that none of its neighbours can make geographical progress along the itinerary. If C chooses B as the target node in the recovery mode, B then enters basic mode as C is closer to itinerary than B. To avoid this problem, we denote the nodes which have been visited as the lowest priority in the forwarding table, which means these visited nodes can be only visited again under the condition that other nodes have no unvisited neighbour. In the example, C will forward the query message to F instead of B and F will enter basic mode of IKNN to continue the routing.

4.2. Simulation Setup

In this section, we conducted a wide range of simulations to evaluate the performance of MAA and compared it to IKNN. We implement our algorithm with JBotSim simulator (Casteigts, 2013). JBotSim relies upon JDK 1.7 framework. The IDE tool is Eclipse.

System parameters and settings are summarized in Table 1. We generate a network with coverage of $500m \times 500m$. The number of robots deployed in this area is between 250 and 500. Each robot maintains a communication radius between $40m$ and $90m$. The query point is chosen randomly with a k value between 2 and 10. For simplicity, we assume that the message transmission between robots does not fail. We assume that the *MessageDelay* is $20ms$ and processing delay is not considered. To increase the experimental accuracy, the result is based on the average of 100 trials. The energy consumption module in WSRNs is fairly complex and varies from different transmission rate, transmission distance and the size of the package etc. Wang et al. (2006) conducted a research to estimate the energy consumption. The experiment offers a maximum reference value of energy consumption per hop per node in a wireless network as 12J (RF module CC1000 with communication frequency 433MHz). To simplify the evaluation module, in this thesis we will adopt this value to estimate the performance of our algorithm in terms of energy consumption. In IKNN, we denote the width of itinerary

$w = \sqrt{3}r/2$ to optimize the accuracy of the algorithm and total routing length.

Parameter	Definition	Default Value	Range
s	Size of network	$500m \times 500m$	-
R	Communicatio n range	$50m$	$40m$-$60m$
r	Radius of KNN boundary	kdc	-
c	Coefficient to estimate *r*	0.8	0.2-1.4
k	Number of target robot	5	3-10
w	Width of itinerary in IKNN	$\sqrt{3}R/2$	0.3*R*-1.6*R*
e	Robot energy consumption	12*J/messag-e/robot*	-
D	Message delay	20*ms*	-
N	Total number of robots	250	250-500

Table 1: Parameters and Values in Experiment

We define three performance metrics as follows:

- Energy Consumption (*J*): The total energy consumed by message transmission in one single KNN query processing progress.
- Process Latency (*ms*): The average period of time between an event occurs and the result of KNN obtained by the algorithm.

- Process Accuracy (%): The percentage ratio of correct robots reading of *k* nearest robots to the total number of robots reading returned to the Home Robot.

With these parameters, we are able to evaluate the performance of the algorithm and obtain the optimized value of these parameters.

4.3. Simulation Results

In this section, we aim at studying how the various parameters in the algorithm affect MAA and IKNN algorithm and try to obtain the optimal value. We only discuss the impact of one parameter at each subsection and keep the value of other parameters as default.

4.3.1. Impact of Itinerary Width

Recall from Section 2.4.3, based on the theory presented by Xu et al. (2006) that Itinerary width *w* should not be more than $\sqrt{3}R/2$ to ensure the coverage of robots of interest. However, a small itinerary width may enforce the itinerary based algorithm to generate a long itinerary and thus decrease the performance of the algorithm. In this section we will explore the relation between *w* and the performance of IKNN. To ease the evaluation, we denote the communication range of each robot R as $50m$ and let $\frac{\sqrt{3}R}{2} \approx 0.9R = 45m$.

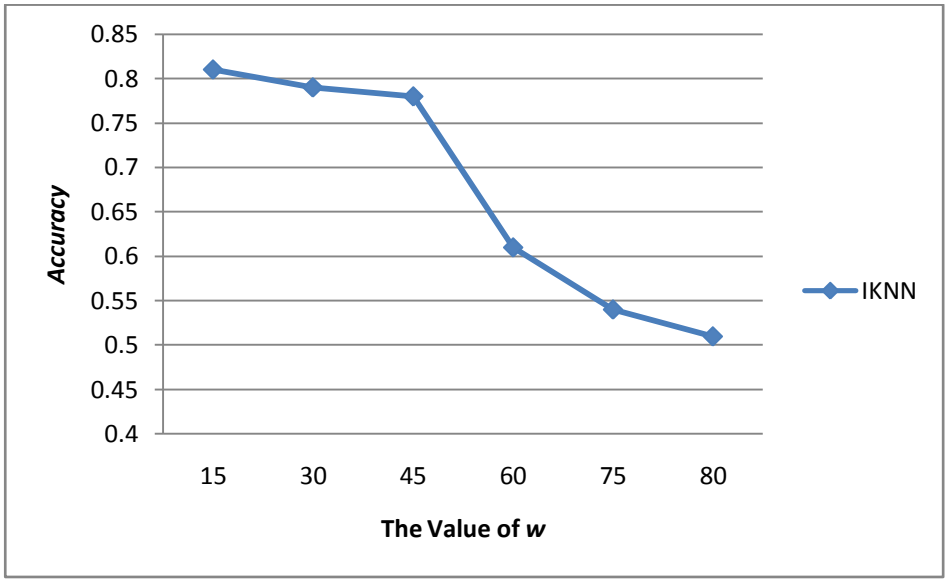


Fig. 20 Impact of *w* on Process Accuracy

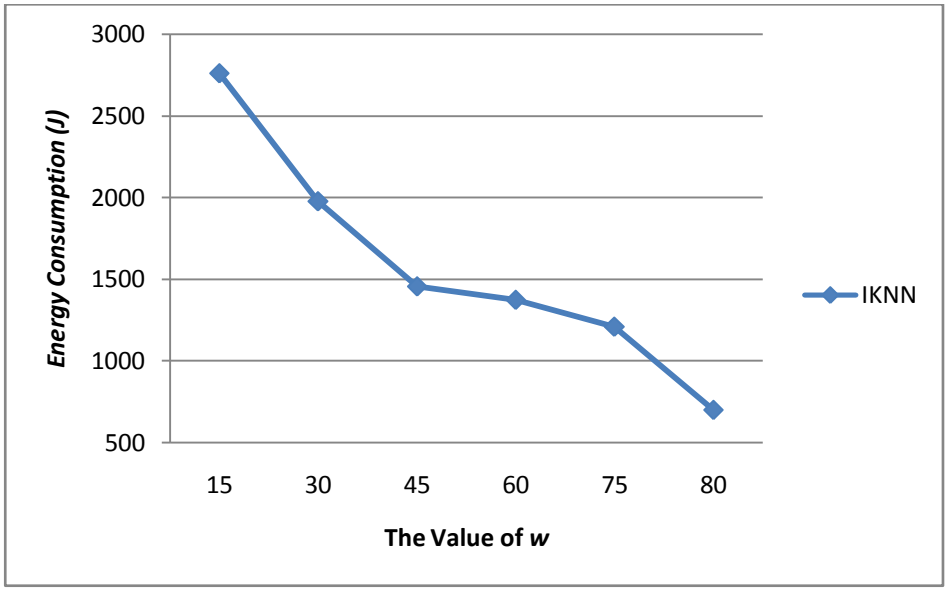


Fig. 21 Impact of *w* on Energy Consumption

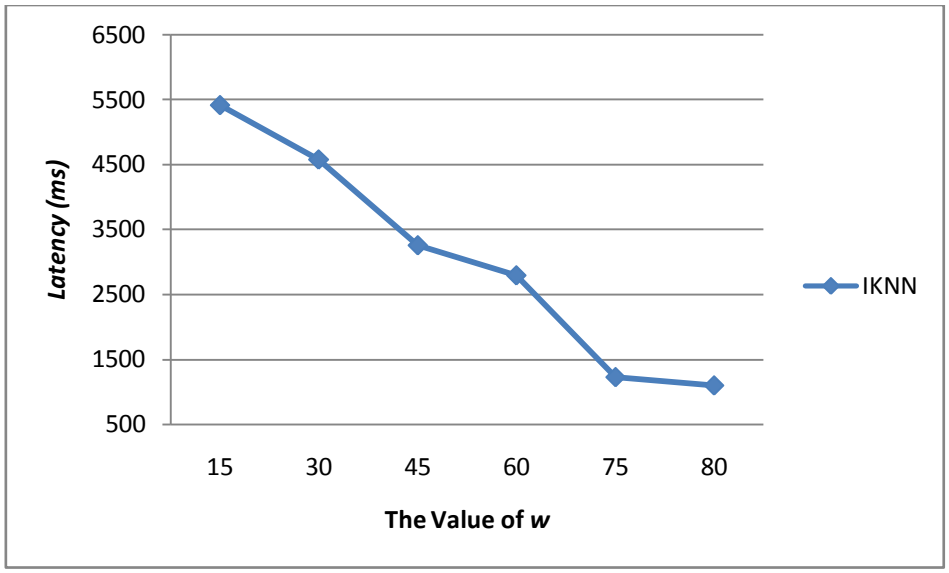


Fig. 22 Impact of *w* on Process Latency

In Fig. 20, Fig. 21 and Fig. 22 the performance of IKNN with different *w* in Process Accuracy, Energy Consumption and Process Latency, respectively, are evaluated. Initially, we notice that the Process Accuracy of IKNN maintains a monotonically decreasing trend as *w* increases. Especially, when *w* is larger than 45*m*, the Process Accuracy drops steeply to an unacceptable level at 0.61. Afterwards, the Process Accuracy maintains a decreasing trend and almost only 50% Process Accuracy is expected when *w* climbs to 80. This is because based on the theory in Xu et al. (2006), after *w* exceeds $\frac{\sqrt{3}R}{2}$, another type of theoretical situation (besides *Itinerary Void*) in which the algorithm may miss robots of interest may occur. Thus, in the case $w > \frac{\sqrt{3}R}{2}$ the Process Accuracy is expected to be very low. A low Energy Consumption and Process Latency are expected as shown in Fig. 21 and Fig. 22 with the ascending

of *w*, respectively. However, this is because a large *w* may enforce the itinerary (in IKNN, the itinerary is the Archimedean spiral) to reach the search boundary and terminate the routing before all interest robots are visited. The experiment result also verifies the same theorem in Xu et al. (2006).

4.3.2. Impact of KNN Boundary

The radius of KNN boundary r plays an important role in MAA, PDFS and IKNN algorithms. Recall from Section 3.1, in our boundary estimation algorithm, r is denoted by a linear equation $r = kdc$, where c is adjustable coefficient and d is the average hop distance from the Home Robot to its neighbours. To optimize the value of c, we denote $k = 5$ and then study the impact of c on the Process Accuracy, Process Latency and Energy Consumption. Thus c can be determined by choosing a balance point among these three metrics.

Our simulation results for this purpose are depicted beginning with Fig. 23.

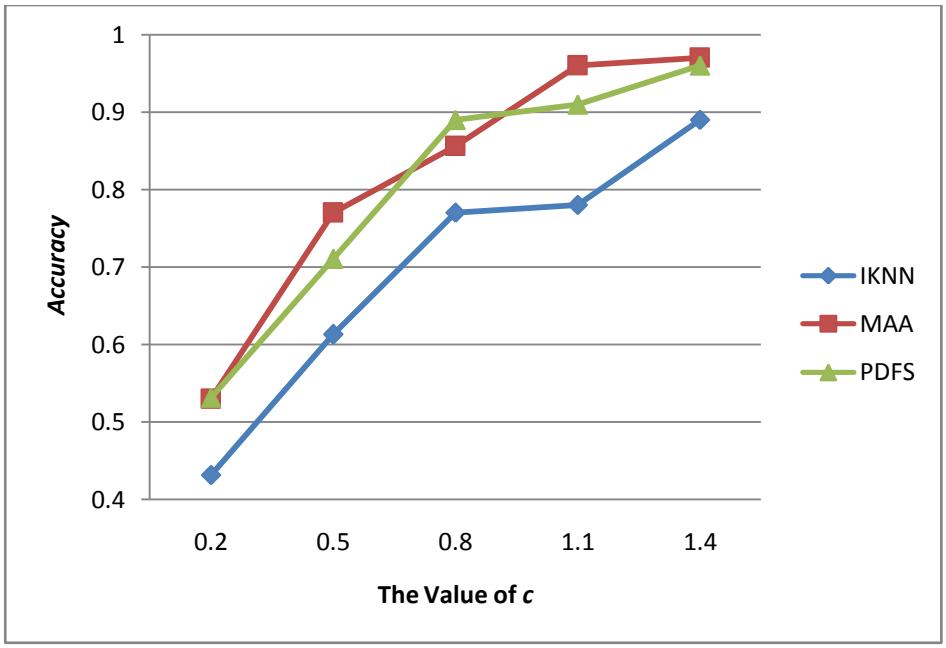


Fig. 23 Impact of *c* on Process Accuracy

Fig. 23 demonstrates the relationship between *c* and Process Accuracy. IKNN, MAA and PDFS experience a fairly low and unacceptable accuracy when c is equals to 0.2. However, then both of them show an increasing trend as *c* is ascending. At the point *c* = 0.8, the rise of IKNN accuracy tends to be smooth and stable with an acceptable accuracy 0.78. Also, the increasing trend of the MAA and PDFS accuracy significantly slows down after the point ***c*** = 1.1 with Process Accuracy = 0.97. Generally, the performances of MAA and PDFS significantly outweigh the performance of IKNN with the same value of c. This gap tends to be narrower after the point *c* =1.1.

An intuitive explanation to Fig. 23 is that the larger the KNN boundary is, the less possibility that the nearest robot is missed. For

MAA and PDFS, with a larger KNN boundary, it is more likely that all the robots around query point are traversed at the end of the algorithm. For IKNN, a small search boundary may enforce the routing being terminated with a short itinerary which may miss multiple target robots on the opposite side of the starting point of the itinerary.

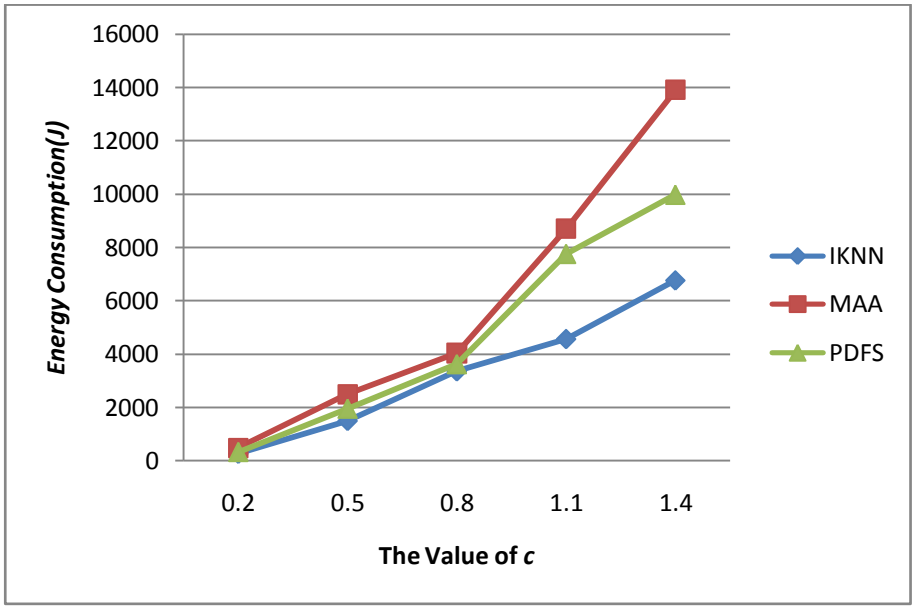


Fig. 24 Impact of *c* on Energy Consumption

The relationship of Energy Consumption and *c* is presented in Fig. 23. These two algorithms are both consuming a relatively small amount of energy with $c = 0.2$. Then the Energy Consumption steadily increases until $c = 0.8$. After this point, the Energy Consumption of IKNN continues to smoothly increase but the Energy Consumption of MAA shows a sharp rise and further increases to the amount twice as much as IKNN. Also, the curve of PDFS increases significantly. We notice that

with this trend the curve of MAA may climb dramatically in the situation that c>1.4.

This trend can be expected because a larger c means a larger KNN boundary and more robots of interest. However, as an Auction Aggregation based protocol, MAA suffers more message transmission when the search area is large. Also, a larger search area brings on more duplicate visit in PDFS. Thus, IKNN outperforms MAA and PDFS in terms of Energy Consumption and could be more energy-saving as c increases.

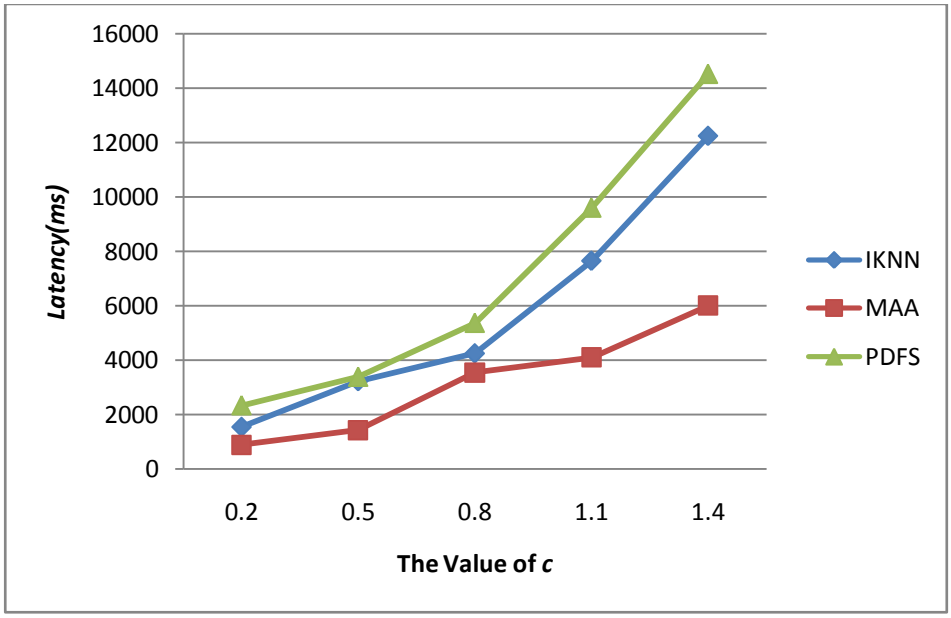


Fig. 25 Impact of *c* on Process Latency

Fig. 23 illustrates the relation between Process Latency and c. We notice an ascending trend in IKNN, MAA and PDFS curves. The Process Latency of IKNN is larger than MAA and PDFS initially but the gap is

slowly becoming narrower. When c = 0.8, IKNN and PDFS curves experience the smallest gap with MAA curve but then steeply increase. The Process Latency of IKNN reaches 12000ms which is twice as that of MAA when c = 1.4. We can expect the IKNN and PDFS curves to further significantly rise with a larger c. MAA curve maintains an approximately linear growth with the explanatory variable c. Generally, MAA outperforms IKNN and PDFS in terms of Process Latency and this superiority is expected to be larger with larger c.

Recall from Section 3.2, MAA is a multithread algorithm and generates multiple copies of query message during the data aggregation progress to search for k nearest robots. In MAA, the data is gathered in parallel by different threads of query message but IKNN and PDFS can only aggregate the data with routing query message as a mono thread algorithm. Thus, IKNN obtains a less satisfactory performance in terms of Process Latency compared to MAA due to a long itinerary when c increases. To conclude, MAA and PDFS give a better Process Accuracy compared to IKNN, but MAA generates tremendous Energy Consumption and DPFS experiences a larger latency. These drawbacks tend to be even worse as the KNN boundary increases. However, MAA and PDFS outperform the IKNN in terms of Process Latency as c increases. In order to optimize the relation between Process Accuracy, Energy Consumption and Process Latency, we consider that the accuracy of both IKNN and MAA reaches a satisfactory level (over 75%) when c

is equal to 0.8. We also notice that although the Process Accuracy of both IKNN and MAA maintains an increasing trend when *c* is larger than 0.8, MAA suffers dramatic energy consumption after that. For reasons stated above, we choose c = 0.8 as a balance between three parameters.

4.3.3. Impact of Number of Target Robot

The main purpose of our algorithm is to find *k* nearest robots to the query point. Thus the impact of the number of target robots (i.e. *k*) on the performance of the algorithm should be drawn attention to. Particularly, whether the algorithms can operate efficiently and generate less Energy Consumption and Process Latency with various value of *k* is a significant evaluation criterion. For this purpose, we conduct several experiments to explore the relation between *k* and the performance of MAA, IKNN and PDFS. In this section, we choose c = 0.8 to estimate the KNN boundary with the equation $r = kdc$.

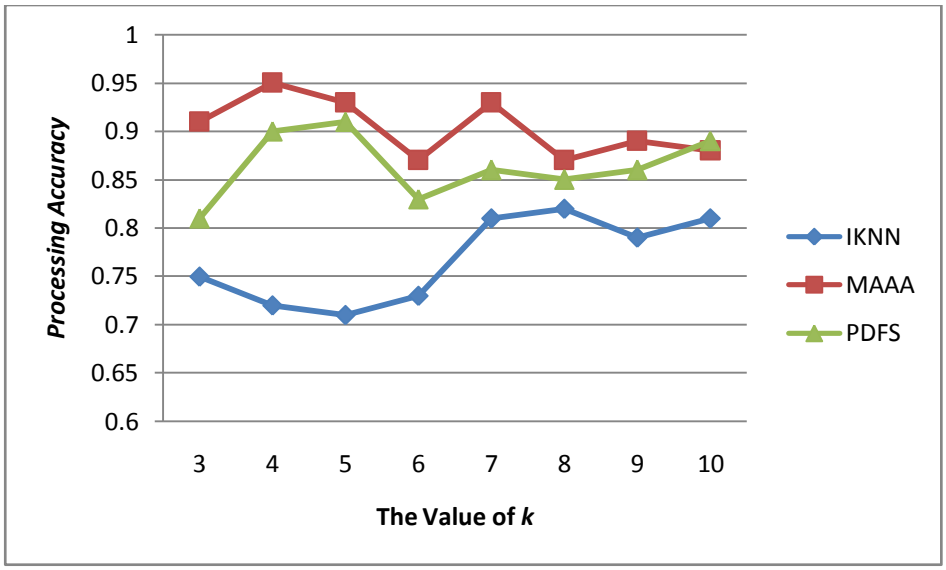


Fig. 26 Impact of *k* on Process Accuracy

Fig. 26 shows the impact of *k* on Process Accuracy. We notice that IKNN curve experiences a decreasing trend when *k* = 3, 4, 5 and reaches the bottom at *k* = 5 and then increases slowly. Afterwards, the IKNN curve tends to level off and fluctuates around the value of 0.8. The Process Accuracy of MAA and PDFS experiences a satisfactory increasing trend and peaks at 0.95, 0.93 respectively when *k* = 4. Then MAA and PDFS curves decline slightly and fluctuate around a value of 0.9.

Recall from Section 3.1 that $r = kdc$ and thus a larger *k* indicates a larger KNN boundary with a given *c*. However, on the contrary, for a given region, a larger *k* also presents a larger probability that there are not enough robots included within this area which lowers the Process Accuracy. Ideally, the algorithm should adjust search boundary dynamically to adapt to the change of *k* as well as maintain an acceptable

accuracy. We notice that as *k* increases, the Process Accuracy of IKNN, MAA and PDFS fluctuates around an acceptable value, which indicates that both algorithms operate reliably with different *k*. We also notice that the overall performance of MAA and PDFS noticeably outweighs that of IKNN in terms of Process Accuracy.

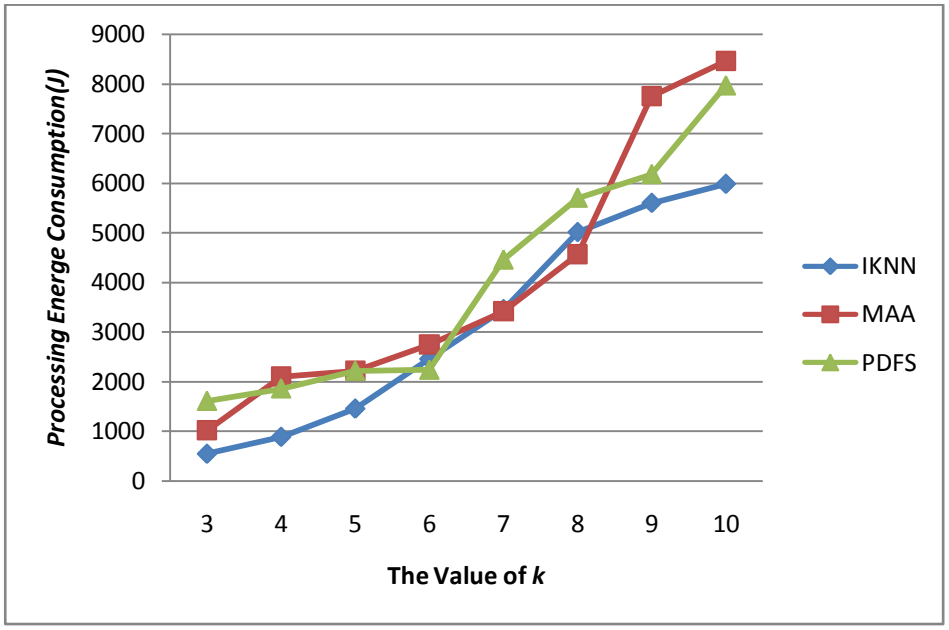


Fig. 27 Impact of *k* on Energy Consumption

Fig. 27 illustrates the impact of *k* on Energy Consumption for IKNN, MAA and PDFS. We notice that the three algorithms' curves experience a monotonically increasing trend as *k* increases. Particularly, IKNN initially consumes less energy when *k* is less than 6 and experiences a more stable ascending than MAA until *k* = 8. At this point, IKNN consumes even more energy than MAA but afterwards MAA curve

sharply increases and the gap between these two algorithms tends to become larger.

Naturally, an expanded KNN boundary indicates that more robots are included within this region. Thus, by visiting these robots and aggregating their data, more Energy Consumption is expected. Note that the Energy Consumption of MAA and PDFS increases far more rapidly than IKNN as KNN boundary expands, which addresses the same conclusion with Section 4.3.2.

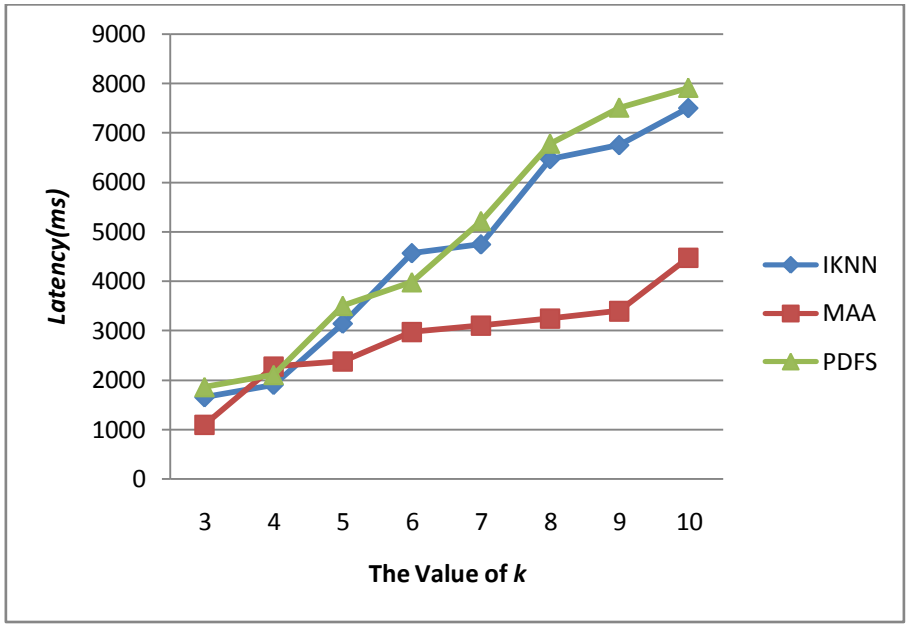


Fig. 28 Impact of *k* on Process Latency

Fig. 28 demonstrates the impact of *k* on Process Latency. The three algorithms show an increasing trend as *k* ascends. Generally, MAA and PDFS curves rise more moderately than IKNN curve and MAA constantly obtains a low Process Latency except when $k = 4$, where

IKNN encounters a lower Process Latency than MAA and PDFS. The gap between the two algorithms becomes wider as *k* rises.

The increase of *k* leads to the expanding of KNN boundary, which also indicates a longer itinerary. Thus, the Process Latency of IKNN increases significantly. Also, more robots are traversed by PDFS. Due to the fact that MAA is less affected by the KNN boundary expanding, MAA keeps a better performance compared to IKNN and PDFS on Process Latency.

To conclude, PDFS, IKNN and MAA maintain a satisfactory performance as *k* increases. The drawback of MAA is that it continues to consume a larger amount of energy than IKNN with a fixed *k*. PDFS and IKNN suffer a larger Process Latency than MAA with different *k*.

4.3.4. Impact of Network Density

In this section, we present several experimental results to verify the relation between network density and the performance of the two algorithms. To achieve this goal, the size of network area is fixed at $500m \times 500m$ and different number of robots (denoted as *N*) is deployed within this area to evaluate the two algorithms.

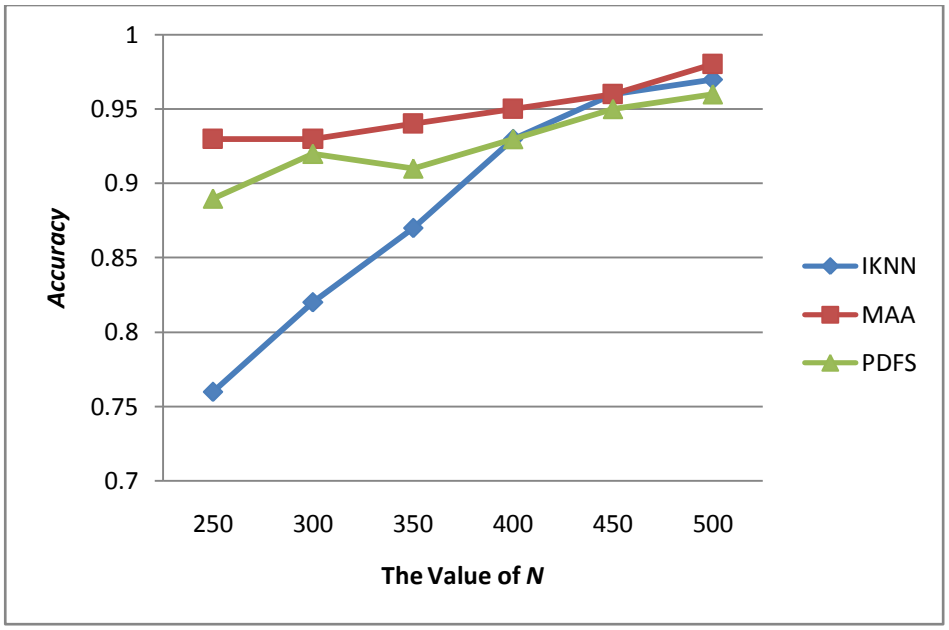


Fig. 29 Impact of *N* at Process Accuracy

Fig. 29 indicates the impact of *N* on the Process Accuracy of IKNN and MAA. IKNN, PDFS and MAA show an upward trend as *N* increases. We notice that IKNN experiences a fairly low Process Accuracy initially at the start and recovers as *N* ascends. MAA maintains a relatively high Process Accuracy compared to IKNN but this advantage is gradually reduced with the rising of *N*. When $N = 450$, the two algorithms obtain approximately the same Process Accuracy.

This trend can be explained by the basic mechanism of IKNN. Recall from Section.4.1, that the *Itinerary Void* is the essential factor resulting in the inaccuracy in IKNN (Xu et al. 2006). Commonly, a dense network is supposed to be more likely to avoid the *Itinerary Void* (Xu et al. 2006). As a result, with the rising of network density, the possibility to encounter *Itinerary Void* is decreased and thus higher

Process Accuracy is expected. MAA and PDFS also perform better with a denser network as the more robots are deployed in KNN boundary, the larger possibility that all robots around query point is traversed which indicates a higher Process Accuracy. However, this trend is not as significant as IKNN.

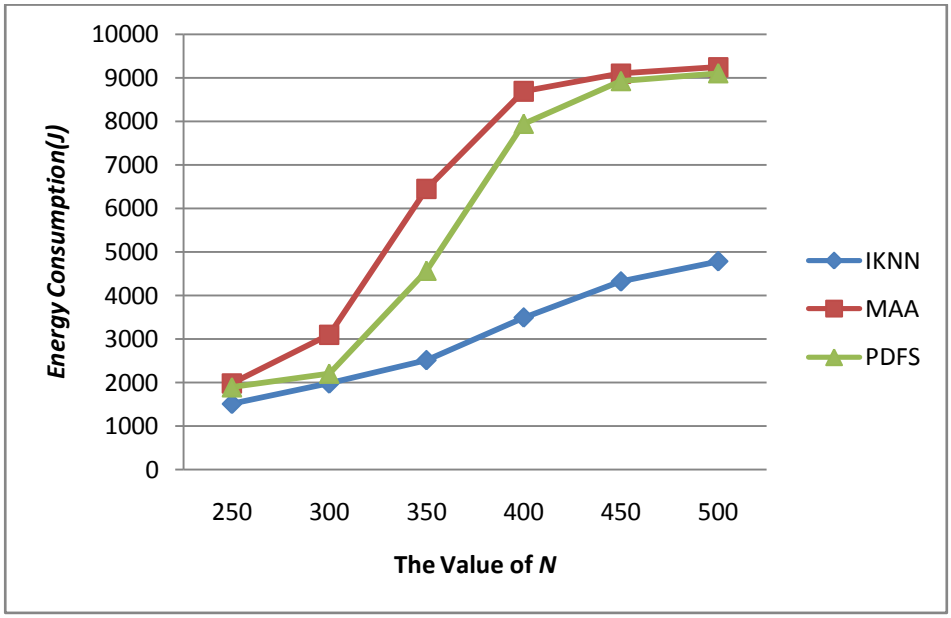


Fig. 30 Impact of *N* on Energy Consumption

Fig. 30 shows the impact of *N* on Energy Consumption of two algorithms. Again, for both algorithms we notice a monotonically increasing Energy Consumption as *N* increases. IKNN curve ascends moderately and constantly keeps a lower Energy Consumption level than MAA. The MAA curve dramatically rises to a significant level as *N* increases and consumes approximately twice energy as that of IKNN when N =500.

The difference in performance between these two algorithms is similar to the conclusion we presented in Section 4.3.1. With more robots in the search region, this Auction Aggregation based algorithm suffers more message transmission because this multiple threads algorithm may visit multiple robots outside of the search region before query message backtracking. The number these non-object robots increases with the ascending of boundary or network density which leads to extra Energy Consumption. However, mono thread algorithms, PDFS and IKNN only visit one non-object robot before backtracking. Thus, IKNN and PDFS are less affected by the expanding of KNN boundary of the increasing of network density.

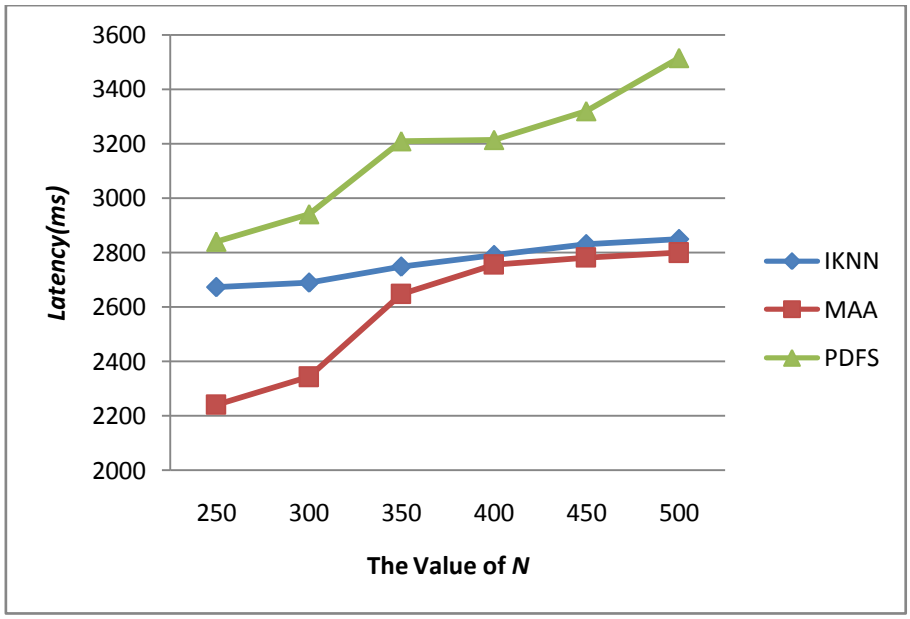


Fig. 31 Impact of *N* on Process Latency

Fig. 31 illustrates the relation between N and Process Latency. We notice that when the number of robots in the network is small, MAA maintains a lower Process Latency than IKNN and PDFS. However, with the increasing of *N*, the gap between two algorithms becomes narrower. Particularly, the Process Latency of IKNN is very close to that of MAA when *N* is larger than 400. Generally speaking, these two algorithms experience an upward trend in Process Latency.

A dense network indicates that for a fixed search area, there should be more robots deployed in the area. In MAA, more robots will be visited to gather the data. In IKNN, a longer itinerary is expected with a dense network. Both situations will lead to the rising of Process Latency. In PDFS, more robots are visited before the traversal is completed.

To conclude, a dense network enhances the performances of IKNN, MAA and PDFS. Particularly, IKNN obtains a fairly high Process Accuracy which is close to the one of MAA when *N* is over 400. MAA and PDFS again generate a relatively large amount of energy as *N* increases but obtains a low level of Process Latency.

4.3.5. Impact of Robot Communication Range

Onat et al. (2008) verified that the communication range of robots indicates the possible number of the neighbours of the robots, which may affect the performance of IKNN, MAA and PDFS. In this section we

study the relation between the communication range of the robot and the performance of these three algorithms.

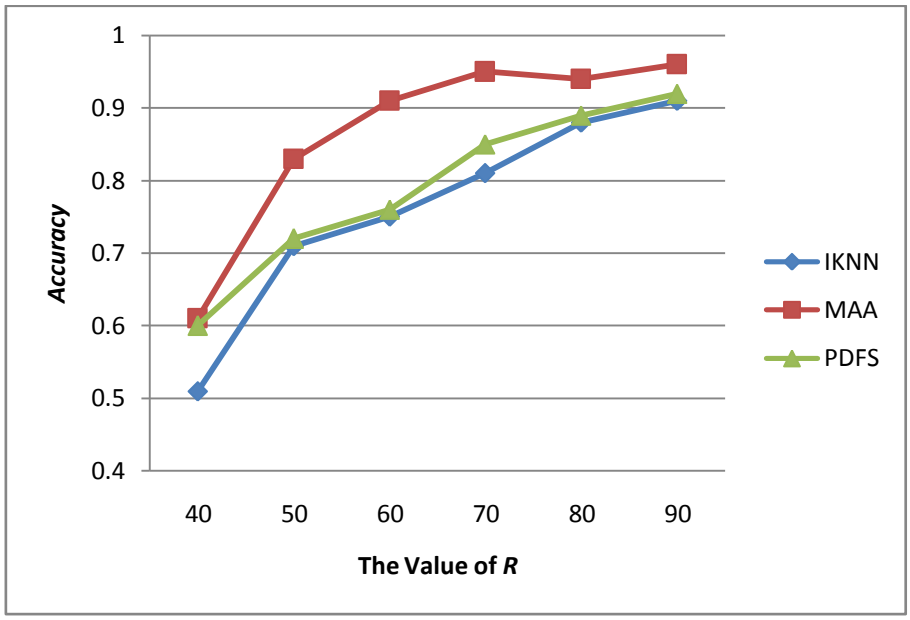


Fig. 32 Impact of *R* on Process Accuracy

In Fig. 32, initially, we observe the relation between *R* and the Process Accuracy of the two algorithms. PDFS, MAA and IKNN show an increasing trend as *R* increases. We notice that these three algorithms obtain an unacceptable Process Accuracy when *R* is 40 and then MAA curve recovers to a relatively high level. The increasing trend of MAA curve becomes smooth when *R* is more than 80.

Theoretically, a small *R* indicates the high possibility to obtain limited number of neighbours for particular robot (Onat et al. 2008). Thus, fewer paths are likely to be generated and the possibility to encounter *Itinerary Void* increases in IKNN. In MAA and PDFS the limited number of paths

between robots may let these two algorithms miss more robots of interest. As a result, a communication of 40*m* is not an acceptable value.

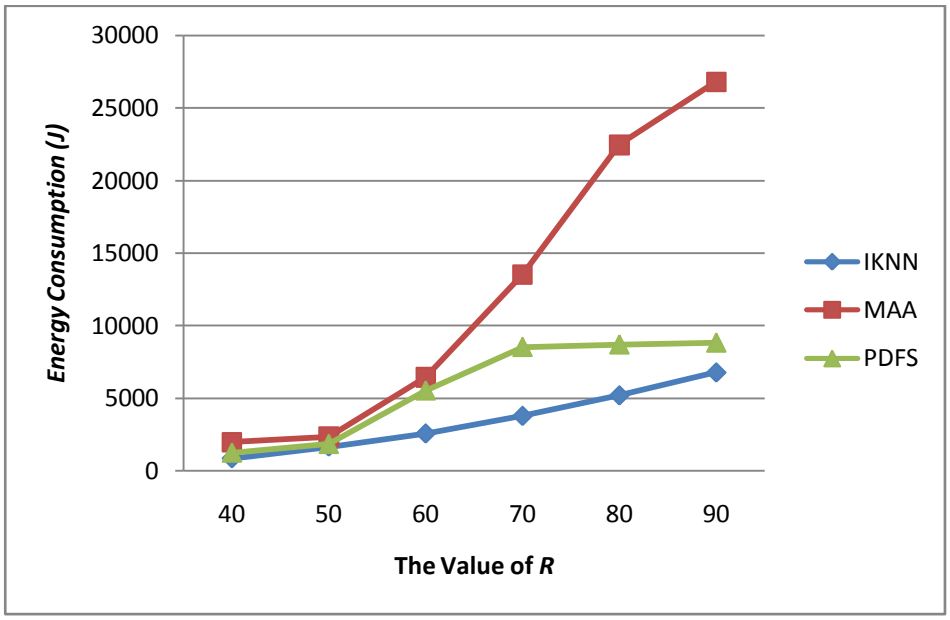


Fig. 33 Impact of *R* on Energy Consumption

The relation between *R* and Energy Consumption is presented in Fig. 33. Both IKNN curve and MAA curve maintain an upward trend as *R* increases. Particularly, the MAA curve begins to dramatically ascend when *R* is larger than 50*m* and the gap between IKNN and MAA in Energy Consumption is getting wider with the increase of *R*.

This trend can be expected with a larger number of neighbours. Both IKNN and MAA have to communicate with more robots when visiting the robots of interest and thus lead to extra Energy Consumption. However, PDFS is less affected and maintains a relatively low Energy Consumption.

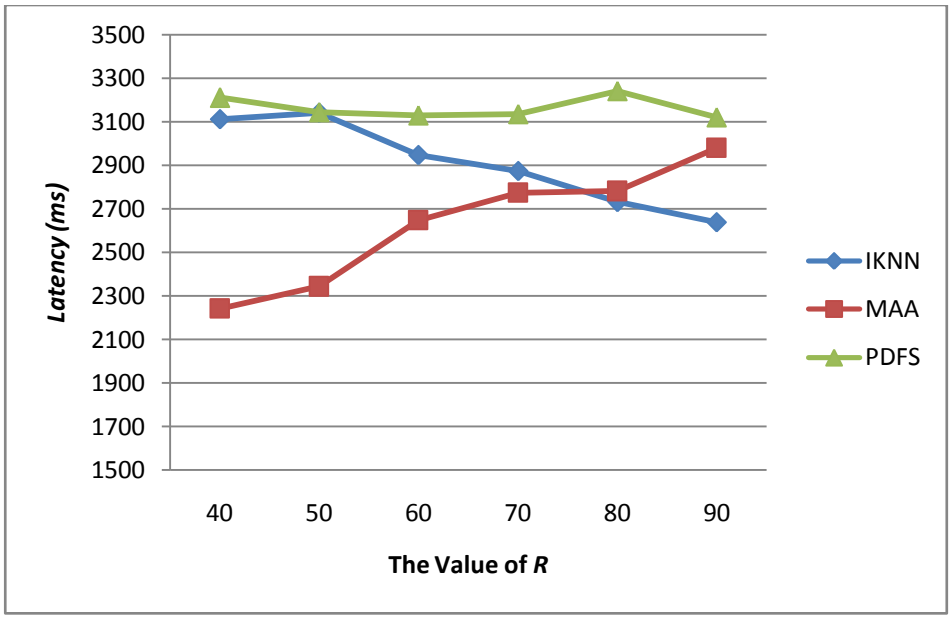


Fig. 34 Impact of *R* on Process Latency

Fig. 34 illustrates the relation between R and Process Latency. We notice the trends in IKNN curve and MAA curve are opposite. MAA curve experiences a gradual upward trend while IKNN curve maintains a decreasing trend. When R is over 80m, IKNN obtains less Process Latency than MAA. While PDFS curve maintains stable and fluctuates around 3100ms.

Recall from Section 4.2, the equation $w = \sqrt{3}R/2$ indicates that a larger R is expected to result in a wider itinerary. Considering the conclusion stated in Section 4.3.4, the increase of R should further enable a shorter itinerary which reaches the KNN boundary fast. This brings impact on two aspects. The Process Accuracy should keep a low level due to inadequate searches for robots of interest around the query point but on the contrary more neighbours are expected which also improve the

accuracy. This is the main reason that in Fig. 32 the Process Accuracy of IKNN maintains a low level as *R* increases. The second impact of increasing *R* is that the query message is expected to be forwarded back to Home Robot because of the short itinerary as presented in Fig.32. On the contrary, for MAA, the increasing of the number of neighbour results in more message exchange and with fixed KNN boundary, more Process Latency is expected.

Chapter 5: Conclusions and Future Work

In this thesis the problem of k nearest neighbour query processing with multiple solutions in wireless sensor and robot network was addressed. We first presented a novel idea to estimate the KNN boundary based on the partial data around the query point. We proposed a linear equation $r = kdc$ to estimate the KNN boundary and then proved its rationality. In query dissemination phase, we presented three possible approaches to address the message dissemination and aggregation. The first approach was called *Multiple Auction Aggregation* (MAA) algorithm which was based on auction protocol. This multithread algorithm constructed a tree rooted at Home Robot and disseminated multiple query messages into the network to get the best biddings of the children. The second approach was a depth first based algorithm with partial search, namely *Partial Depth First Search* (PDFS). With a low protocol complexity, this approach attempted to traverse all the robots encircled by KNN boundary and aggregate their data.

In the simulation part, we first optimized the Itinerary based KNN algorithm (IKNN) as a comparative group. We presented two methods to lower the possibility of encountering *Itinerary Void* in IKNN. Then we compared IKNN with MAA in Process Accuracy, Energy Consumption and total Process Latency. As stated in Section 4.3, based on our experimental results, the Process Accuracy of MAA and PDFS far

outperform Process Accuracy of IKNN in almost all occasions. However, MAA suffered larger Energy Consumption than IKNN especially when the KNN boundary was expanded or network density was increased. Also PDFS experienced a larger latency than other two algorithms. Considering the facts that the message transmission is operated in robot network layer and robots always carry a much larger energy source than sensors, this drawback is not significant in most occasions. Also, the Process Latency of MAA is relatively low compared to IKNN and thus offers a quick reflection in emergency occasions. As a result, both MAA and PDFS are satisfactory algorithms to solve KNN query processing in Wireless Sensor and Robot Networks.

Further improvement can be made to optimize the Energy Consumption of MAA.

References

[1] Akyildiz, I. F., W. Su, Y. Sankarasubramaniam, and E. Cayirci. "Wireless sensor networks: a survey". *Computer Networks* 38, No. 4 (2002): 393-422.

[2] Akyildiz, I. F., X. Wang, and W. Wang. "Wireless mesh networks: a survey". *Computer Networks* 47, No. 4 (2005): 445-487.

[3] Bose, P., P. Morin, I. Stojmenović, and J. Urrutia. "Routing with guaranteed delivery in ad hoc wireless networks". *Wireless Networks* 7, No. 6 (2001): 609-616.

[4] Bretscher, O., *Linear algebra with applications*. Eaglewood Cliffs, NJ: Prentice Hall, 1997.

[5] Casteigts, A., *The JBotSim library.* CoRR, abs/1001.1435, 2013.

[6] Chang, C-Y., C-T. Chang, Y-C. Chen, and H-R. Chang. "Obstacle-resistant deployment algorithms for wireless sensor networks". *Vehicular Technology, IEEE Transactions on* 58, No. 6 (2009): 2925-2941.

[7] Chempavathy, J., and V. Vijayaraja. "Optimizing parallel concentric circle itinerary based KNN query processing in Wireless Sensor Networks". In *Trendz in Information Sciences & Computing (TISC), 2010*, pp. 226-229. IEEE, 2010.

[8] Demirbas, M., and H. Ferhatosmanoglu. "Peer-to-peer spatial queries in sensor networks." In *Peer-to-Peer Computing, 2003. (P2P 2003). Proceedings. Third International Conference on*, pp. 32-39. IEEE, 2003.

[9] Dias, M. B., R. Zlot, N. Kalra and A. Stentz. "Market-based multirobot coordination: A survey and analysis". *Proceedings of the IEEE* 94, no. 7 (2006): 1257-1270.

[10] Fu, T-Y., W-C. Peng, and W-C. Lee. "Optimizing parallel itineraries for knn query processing in wireless sensor networks". In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pp. 391-400. ACM, 2007.

[11] Fu, T-Y., W-C. Peng, and W-C. Lee. “Parallelizing itinerary-based KNN query processing in wireless sensor networks”. *Knowledge and Data Engineering, IEEE Transactions on* 22, No. 5 (2010): 711-729.

[12] Goldin, D., M. Song, A. Kutlu, H. Gao, and H. Dave. "Georouting and delta-gathering: Efficient data propagation techniques for geosensor networks." *GeoSensor Networks. Boca Raton* (2005): 73-95.

[13] Heissenbüttel, M., T. Braun, T. Bernoulli, and M. Wälchli. “BLR: beacon-less routing algorithm for mobile ad hoc networks”. *Computer communications* 27, No. 11 (2004): 1076-1086.

[14] Ivan M., V. Malbasa, and I. Stojmenovic, “Robot to robot: communication aspects of coordination in robot wireless networks”. IEEE Robotics & Automation Magazine, Vol. 17, no. 4, pages 63-69, 2010.

[15] Jayaraman, P. P., A. Zaslavsky and J. Delsing. “Cost-efficient data collection approach using k-nearest neighbors in a 3D sensor network”. In *Mobile Data Management (MDM), 2010 Eleventh International Conference on*, pp. 183-188. IEEE, 2010.

[16] Karp, B. and H-T. Kung. “GPSR: Greedy perimeter stateless routing for wireless networks”. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pp. 243-254. ACM, 2000.

[17] Kim, J-J., J-J. Kang, K-Y. Lee, G-S. Choi, Yong-Soon Im, and E-Y. Kang. “Efficient Processing of KNN Queries in Wireless Sensor Networks”. *IJSEA Journal*, 7, No. 2, (2013):137-148

[18] Komai, Y., Y. Sasaki, T. Hara and S. Nishio. “A kNN query processing method in mobile ad hoc networks”. In *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*, vol. 1, pp. 287-288. IEEE, 2011.

[19] Kumar, V., D. Rus and S. Singh. “Robot and sensor networks for first responders”. *Pervasive Computing, IEEE* 3, No. 4 (2004): 24-33.

[20] Morin, P. R. "Online routing in geometric graphs." PhD diss., Carleton University, 2001.

[21] Onat, F. A., I. Stojmenovic, and H. Yanikomeroglu. “Generating random graphs for the simulation of wireless ad hoc, actuator, sensor, and internet networks”. *Pervasive and Mobile Computing* 4, No. 5 (2008): 597-615.

[22] Roussopoulos, N., S. Kelley and F. Vincent. “Nearest neighbor queries”. *ACM sigmod record* 24, No. 2 (1995): 71-79.

[23] Samet, H. “Depth-first k-nearest neighbor finding using the MaxNearestDist estimator”. In *Image Analysis and Processing, 2003. Proceedings. 12th International Conference on*, pp. 486-491. IEEE, 2003.

[24] Sanchez, J., P. Ruiz, and R. Marin-Perez. “Beacon-less geographic routing made practical: challenges, design guidelines, and protocols”. *Communications Magazine, IEEE* 47, No. 8 (2009): 85-91.

[25] Song, Z., and N. Roussopoulos. "K-nearest neighbor search for moving query point." In *Advances in Spatial and Temporal Databases*, pp. 79-96. Springer Berlin Heidelberg, 2001.

[26] Stojmenovic, I. "Geocasting with guaranteed delivery in sensor networks."*Wireless Communications, IEEE* 11, No. 6 (2004): 29-37.

[27] Wang, G., G. Cao, T. La Porta and W. Zhang. “Sensor relocation in mobile sensor networks”. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, Vol. 4, pp. 2302-2312. IEEE, 2005.

[28] Wang, Q., M. Hempstead and W. Yang. “A realistic power consumption model for wireless sensor network devices”. In *Sensor and Ad Hoc Communications and Networks, 2006. SECON'06. 2006 3rd Annual IEEE Communications Society on,* vol. 1, pp. 286-295. IEEE, 2006.

[29] Winter, J. and W-C. Lee. “KPT: a dynamic KNN query processing algorithm for location-aware sensor networks”. In *Proceedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB 2004*, pp. 119-124. ACM, 2004.

[30] Winter, J., Y. Xu, and W-C. Lee. “Energy efficient processing of k nearest neighbor queries in location-aware sensor networks”. In *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on*, pp. 281-292. IEEE, 2005.

[31] Wu, S-H., K-T. Chuang, C-M. Chen and M-S. Chen. “Diknn: an itinerary-based knn query processing algorithm for mobile sensor networks”. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pp. 456-465. IEEE, 2007.

[32] Wu, S-H., K-T. Chuang, C-M. Chen and M-S. Chen. “Toward the optimal itinerary-based KNN query processing in mobile sensor networks”. *Knowledge and Data Engineering, IEEE Transactions on* 20, No. 12 (2008): 1655-1668.

[33] Xu, Y., T-Y. Fu, W-C. Lee, and J. Winter. “Processing k nearest neighbor queries in location-aware sensor networks”. *Signal Processing* 87, No. 12 (2007): 2861-2881.

[34] Xu, Y.i, W-C. Lee, J. Xu, and G. Mitchell. "Processing Window Queries in Wireless Sensor Networks." In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pp. 70-70. IEEE, 2006.

[35] Yu, X., X. Yu and D. Zhou. “A Depth-first Adaptive KNN Searching Algorithm”. In *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, Vol. 2, pp. 10318-10322. IEEE, 2006.