The Hong Kong Polytechnic University

Department of Computing

Coverage and Data Aggregation for Object Tracking in Wireless Sensor Networks

by

JINGJING LI

A thesis submitted in partial fulfillment of the requirements for

the Degree of Doctor of Philosophy

Jun 2013

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____(Signature)

Jingjing LI (Name of Student)

Abstract

With the development of sensing and wireless communication technologies, more and more applications are designed for wireless sensor networks (WSNs). Object tracking is one of the most important applications, which senses the location and movement of the target and delivers the sensed data to the users. It has been proved challenging to design algorithms, protocols, and systems for object tracking in WSNs. On one hand, wireless sensors have limited sensing range, communication range, storage, energy, and other resources. Therefore, a single sensor node cannot sense the whole area being monitored or store many data. On the other hand, the continuous tracking requires the algorithms and protocols designed for calculating the location of the target and delivering the data to be simple but accurate. Therefore, there is a tradeoff between the accuracy and the complexity of algorithms and protocols.

The process of object tracking can be divided into three stages: deployment stage, detection stage, and tracking stage. There are many research problems in the three stages of object tracking such as sleeping schedule, coverage, routing, data aggregation, prediction of the location of target, etc. The reliability and the efficiency of the solution are limited by energy, sensing range, and storage capability. In this thesis, we research on the challenging issues in designing algorithms to improve the performance of coverage, routing, and data aggregation for object tracking in WSNs. The results of this thesis research can be described in three parts as follows.

In the first part, the sleeping schedule and coverage maximization are studied, answering questions of how to reduce the number of sensors being used and how to enhance the reliability of the network. We focus on how to find the maximum

number of the disjoint cover sets, each can completely cover the given area, and how to identify the corresponding sets of sensors. We develop a localized approach to designing the evolutionary algorithm (EA) in distributed systems. A scheme, named localized evolutionary algorithm (LEA), is proposed to solve optimization problems that can be divided into several subproblems. LEA is different from parallel EA where the global information is required for searching the solution space and consequently, consumes a large amount of time and space resources. In LEA, a set of processors solve the global optimization problem in a cooperative way. First, each processor is assigned to solve a subproblem based on locally collected information. Then the locally optimal solutions obtained by all the processors are combined to obtain the globally optimal or near optimal solution to the whole problem. We analytically discuss the optimal number of subproblems that the problem should be divided and the performance metrics for evaluation the proposed algorithm. Next, we describe the optimization problem that the proposed algorithm that are suitable for being solved by LEA. The characteristics of the suitable optimization problems are also summarized. We apply the proposed LEA to solve the sleeping schedule and coverage in WSNs. Simulations have been carried out to validate the effectiveness of our proposed algorithm in terms of the number of disjoint sets, storage consumed in each node, and calculation time.

In the second part, we study the coverage-preserving routing problem, i.e., how to find a routing path in a WSN with the maximum sensing coverage provided by the nodes on the path of object movement subject to the delay constraint. In most of the WSN applications, covering the area of interest and delivering the sensed information to the sink are two fundamental functions. Extensive research associated with these two issues, such as energy efficient coverage and delay-constraint routing, can be found in the literature. However, few works combine these two issues together. Considering the fact that wireless sensors can take the responsibility of both

sensing and routing, it is expected that a solution jointly considering these two issues will provide more benefit. We first define the coverage-preserving routing problem and prove this problem is NP-hard. Then two coverage-preserving routing algorithms, one centralized and another distributed, are designed. The two algorithms are based on label setting algorithm (LS) and genetic algorithm (GA) respectively. Both algorithms are based on Monte-Carlo integration to approximately calculate the sensing area. Then the sensing area between the nodes is regarded as the weight of edges to find the routing path from source node to sink node. Simulation results show that the proposed algorithms achieve better performance than other previous algorithms in terms of sensing area of routing path as well as the number of hops in the routing path.

In the third part, we apply our studies on dynamic construction of data aggregation tree to track the location of the target. We first consider the sensor set selection scheme in a fully distributed way. Based on the prediction model, the current active sensors wake up the next set of sensors which are around the predicted location of the target independently. After obtaining the set of sensors to be active, the data aggregation tree from the set of sensors to the sink node is constructed quickly and energy efficiently. When the target is moving, the tree will be reconstructed in real time. Simulation results show that the proposed algorithm achieves much better performance than other methods for constructing data aggregation tree, in terms of the number of active nodes involved and the time of construction of the tree.

Publications

Journal Papers

1. **Jingjing Li**, Jiannong Cao, Survey of Object Tracking in Wireless Sensor Networks, *Ad Hoc & Wireless Sensor Network*, Accept.
2. **Jingjing Li**, Jiannong Cao, A localized Approach for Evolutionary Algorithms, *IEEE Transaction on Parallel and Distributed Systems*, submitted.
3. Zhihui Zhan**, Jingjing Li**, Jiannong Cao, Jun Zhang, Henry Shu Hung Chung, and YuHui Shi, Multiple Populations for Multiple Objectives: A Co-evolutionary Technique for Solving Multi-objective Optimization Problems, *IEEE Transactions on Systems, Man, and Cybernetics--Part B: Cybernetics.*, pp.445-463, vol(43), Apr 2013.
4. Weigang Wu, Jiannong Cao, Xiaopeng Fan, **Jingjing Li**, Design and Performance Evaluation of Overhearing-aided Data Caching in Wireless Ad Hoc Networks, *IEEE Transactions on Parallel and Distributed Systems*, Accept.

Conference Papers

5. **Jingjing Li**, Jiannong Cao, Distributed Coverage-Preserving Routing Algorithm for Wireless Sensor Networks, *IEEE International Conference on Communications (ICC)*, Kyoto, pp. 1-5, 5-9 June 2011.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor-Prof. Jiannong Cao for his great patience, helpful guidance, constant encouragement and financial support. Without his comments, suggestions, and contributions, this research work will never be finished. He leads me into the field of wireless sensor network research and let me know how to do research. I learn a lot from him, not only the scientific research methods, but also his esteemed personal attitude towards looking satisfaction in the professional development. I will benefit from those through all my life.

I am also deeply grateful to Dr. Xuefeng Liu for our fruitful collaborations and discussions. We explored ideas and wrote papers together. I will also show my thanks to Dr. Weigang Wu, Dr. Hejun Wu, Weipin Zhu, Jie Zhou, Yang Liu, Yuan Zheng, Jin Xie, Binbin Zhou for their kind and valuable help.

Finally, I would like to express my heart-felt gratitude to my parents, whose unwavering faith and confidence in me is what drives me forward to make this far, and to continue on the road ahead.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CA: Constant Acceleration

CCP: Coverage Configuration Protocol

CPRP: Coverage-Preserving Routing Problem

CT: Constant Turn

CV: Constant Velocity

DKF: Distributed Kalman Filter

EA: Evolutionary Algorithm

EEC: Energy-Efficient Coverage

GPS: Global Positioning System

HSS: Hop-based Sleep Scheduling Algorithm

ILP: Integer Linear Programming

JPDA: Joint Probabilistic Data Association Filter

KF: Kalman Filter

LDCC: Layered Diffusion-based Coverage Control

LEA: Localized Approach for Evolutionary Algorithm

LS: Label Setting Algorithm

LUC: Location-Unaware Coverage Algorithm

OGDC: Optimal Geographical Density Control

PEA: Parallel Evolutionary Algorithm

PEAS: Probing Environment and Adaptive Sleeping

PF: Particle Filter

RIS: Random Independent Sleeping

SGGP: Seed Growing Graph Partition Algorithm

VFA: Virtual Force Algorithm

WCSPP: Weight Constrained Shortest Path Problem

WSN: Wireless Sensor Networks

Chapter 1. Introduction

In this thesis, we aim to overview the existing works, investigate the challenging problems and to propose novel techniques, algorithms and systems for efficient object tracking in Wireless Sensor Networks (WSNs). With respect to the topic of object tracking in WSNs, we mainly discuss three related techniques: coverage, routing, and data aggregation. In this chapter, we gives an general introduction to our research, discussing the features of object tracking in WSNs and the challenges in designing algorithms for object tracking. The motivation of our works is based on the discussion. Moreover, the contributions and the organization of the thesis are summarized.

1.1. Motivation

With significant advances in hardware manufacturing technology, WSNs have gained worldwide attention in recent years. WSNs consist of numerous spatially distributed, small, low-cost sensor nodes, which possess the capabilities of signal sensing, computation, storage, and wireless communication [ASS+02]. The functions of WSNs include: data sensing, processing, and transmission [AY02]. These functions consume the limited energy provided by a non-renewable micro-battery on the sensor nodes. Therefore, developing applications for WSNs faces many challenges, such as 1) how to maximize the lifespan of WSNs as long as possible; 2) how to achieve a trade-off between power conservation and quality of sensing; 3) how to deliver within time constraints so that appropriate observations can be made[S04]; 4) the security of WSNs.

In WSNs, Object tracking is an important application. It can be defined as tracking security attacks [HK09], moving object, and changes in environmental monitoring,

such as pressure [LD11], temperature [NS08], and acoustics [CH03]. Recently, research usually focuses on tracking the location of moving objects.

The basic operations of sensors in object tracking include (1) generating the track information continuously; (2) delivering to a collector in real-time. There are three stages in object tracking process, namely “deployment stage”, “detection stage”, and “tracking stage”. The key problem in first stage is about **how to deploy the minimum number of sensors to achieve the optimal level of sensing and communication coverage**. The key problem in second stage is about **how to guarantee the area coverage and extend the lifetime of the networks.** The key problem in third stage is about **how to minimize the number of sensors and get high tracking accuracy.**

It remains some key issues to enhance the efficiency of object tracking in WSNs. Firstly, how to detect and track the target accurately in real-time. Secondly, how to improve accuracy and save energy. Thirdly, how to design tracking algorithm in distributed way.

1.2. Contributions of the Thesis

The main contributions of this thesis are designing new techniques and algorithms for object tracking in WSNs. As shown in Figure 1.1, our contributions primarily focus on three topics. 1) With respect to energy efficient coverage in deployment stage, we focus on sleep scheduling and coverage, in which the sensors in the networks are divided into disjoint sets. The sets are waked up successively, and in each set, the active sensors can cover the area completely. We propose a localized evolutionary algorithm (LEA) framework to find the maximum number of the disjoint cover sets each of which can cover the area completely, and to identify the corresponding sets of sensors. The LEA can also solve the optimization problem that

can be divided into subproblems, in which there is not an exact sequence between the subproblems and the subproblems have the same optimization objectives. 2) With respect to routing in detection stage, we focus on coverage preserving routing, in which a routing path is selected in a WSN with the maximum sensing coverage provided by the nodes on the path. We address the coverage preserving routing problem in object tracking with the constraint of time delay. 3) With respect to data aggregation in tracking stage, we focus on constructing data aggregation tree dynamically, in which the aggregation tree is constructed and reconstructed according to the changes of location of the object.

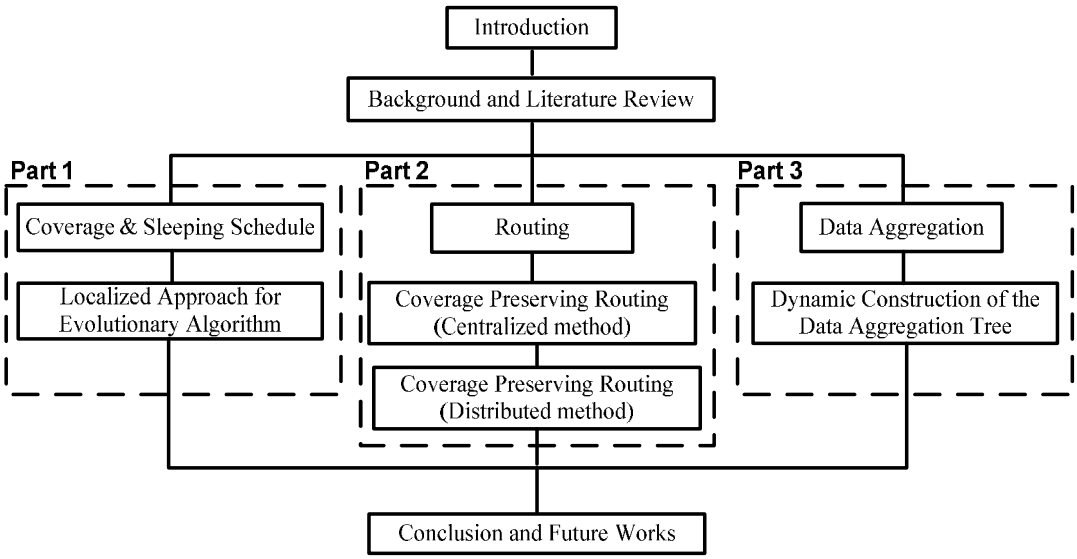

Figure 1.1: An outline of the contributions in this thesis

1.2.1. Contributions in Coverage and Sleeping Schedule

Evolutionary algorithms (EAs) are search techniques which simulate the process of biological evolution [FF95][Z96]. The objective of the algorithm is to find optimal or near optimal solutions to optimization and search problems. The basic mechanisms of EAs consist of recombination, mutation, and selection. The candidate

solutions to the optimization problem are regarded as individuals in a population that are initialized randomly, and the survival individuals are determined by fitness function. The population evolves after the genetic operations are repeated. The final solution is the individual with highest value of fitness. The significant advantages of EAs are conceptual simplicity, broad applicability, parallelism, and robustness. Because of the strengths, EAs have been widely applied to various research fields, such as logistics [MKC09], networking [JW04], scheduling [GMR08], image processing [YT03], pattern recognition [JRM00], data mining [IY04], etc.

In this thesis, a localized EA (LEA) framework is proposed, in which a set of processors solve the global optimization problem in a cooperative way. First, each processor is assigned to solve a subproblem based on locally collected information. Then the locally optimal solutions provided by the parallel processors are combined to obtain the globally optimal or near optimal solution to the whole problem. With our LEA, the energy efficient coverage problem in WSNs can be solved. The details are described in the following subsections.

1.2.1.1. LEA: A Localized Approach for Evolutionary Algorithms

We aim to solve optimization problems in large distributed system, such as in WSNs. Some existing EAs assume that each processor has the prior knowledge of global information. Global information is the information required for obtaining the global solution. Because collecting the global information is time-consuming if the corresponding information is large and distributed on many processors, the scale of the network raises serious matters, especially in the very large systems. Firstly, we propose LEA for solving optimization problems that can be divided into several subproblems. These subproblems have the same optimization objectives and can be solved in parallel way. Then, we discuss the number of search subspaces and evaluation metrics of the LEA and present the suitable applications. The proposed

LEA is applied to the energy-efficient coverage and sleeping schedule problem in WSNs. We also investigate the performance of the proposed algorithm and analyze the results.

1.2.2. Contributions in Coverage Preserving Routing

Considering the fact that a node in a WSN has the responsibility of sensing and data delivery, combining sensing coverage together with routing will have some extra benefits. For example, if the overall sensing area of the nodes in a routing is maximized, less number of remaining nodes in the area needs to be functioned to cover the area. The energy consumption of whole network can therefore be saved. Another advantage of a larger sensing area of the nodes in the routing is that the probability that a new event will be detected by these relay nodes is increased. The new event detected by these relay nodes can be directly sent to the sink along the path without the necessity of establishing a new one. For example, in monitoring of fire in a forest, if the newly developed fire point is covered by the sensor nodes in a routing path, the newly sensed data can be sent along the existing path quickly without finding a new path.

Although both sensing coverage [MKP+05] and routing [AK04] have been studied extensively, there still lacks of researches combining these two issues together. In this thesis, we mainly focus on the coverage preserving routing problem. We proved that this problem is NP-hard and then proposed two methods to calculate the coverage area of the routing based on Monte-Carlo integration method and find the routing path in both centralized way and distributed way. There are three contributions on this topic. Firstly, we propose an efficient and light-weighted method to calculate the total sensing area of multiple nodes, particularly when the sensing areas overlap with each other. Secondly, we propose two algorithms to maximize the sensing coverage of routing under delay constraint. Thirdly, the

proposed algorithms do not need the location information of node. The performance of our algorithm is demonstrated through simulation data. The details are described in the following subsections.

1.2.2.1. A Centralized Method for Coverage Preserving Routing

We consider the coverage preserving routing problem in WSNs, investigating how to find the path from source to sink with maximum sensing coverage while the hop of the path is less than *K*. We propose a centralized algorithm based on genetic algorithm (GA) for coverage preserving routing problem. According to the characteristics of the problem, the operations of GA are modified to select the path. Repairing method and CHECK scheme are designed to repair and check the invalid path. Simulation results show that the proposed centralized algorithm achieves better performance than the existing method.

1.2.2.2. A Distributed Algorithm for Coverage Preserving Problem

Since there are many types of overlapping area of different nodes, it is impossible to compute the sensing area of routing accurately by only using the directional and distance information. Thus, a preprocessing method is proposed to approximately calculate the areas. The method is derived from Monte-Carlo integration in [KW08] which employs some random points to estimate the area. Based on the preprocessing method, we develop a distributed algorithm to solve coverage preserving algorithm, which is modified from label setting algorithm (LS) [AMO93].Analysis and simulation results show that under the same time delay constraint, the proposed algorithms can find a routing path with significantly larger sensing coverage (more than 87% in our simulation) than that obtained considering only hop constraint.

1.2.3. Contributions in Data Aggregation

Data aggregation is the fundamental technique that the sensor collects the sensing data from the source nodes and delivers the information to the sink node for object tracking in WSNs. Construction of data aggregation tree is an interesting method for data aggregation in object tracking in WSNs. Data aggregation tree means sink node is regarded as root and the source nodes are regarded as leafs, there exists efficient paths from the source nodes to the sink through some intermediate nodes that allow in-network consolidation of redundant data.

In this thesis, we propose an algorithm consisting of two components: 1) sensor set selection based on the prediction model and 2) dynamic aggregation tree construction based on the energy consumption model. Many research associated with these two issues, such as information-based selection and tree-based data aggregation, can be found in the literature. However, 1)few works combine these two issues together; 2)as to the selection scheme, most of existing works are not fully distributed that will consume communication cost; 3) as to the construction of aggregation, the existing works do not consider the dynamic value of residual of the each node. Compared with the existing works, we have the following contributions: 1) We propose a localized algorithm that the next sensor is woken up by the active sensor independently. 2) We propose an algorithm to maximize the residual energy of the tree with minimized number of nodes from the source nodes to sink node. 3) The proposed algorithm is fully distributed and does not need the location information. The details are introduced as follows.

1.2.3.1. Construction of Dynamic Data Aggregation Tree

We consider the data aggregation problem for object tracking in WSNs, investigating how to dynamical build a data aggregation tree in WSNs with the minimum number

of nodes in the tree. At the same time, the minimum residual energy of the node in the tree is maximized. We first consider the sensor set selection scheme in fully distributed way. Based on the prediction model, the active sensors at current time slot wake up the next set of sensors which are around the predicted location of the target independently. After obtaining the set of sensors to be active, the data aggregation tree from the set of sensors to the sink node is constructed timely and energy efficiently. When the target is moving, the tree will be reconstructed as soon as possible. Simulation results indicate that the proposed algorithm obtains better performance than other method for constructing data aggregation tree, in terms of the number of active node involved and the time of construction of the tree.

1.3. Outline of the Thesis

The structure of this thesis is described in Figure 1.1.

Chapter 1 provides an introduction to this thesis study.

Chapter 2 presents the background and literature review of object tracking in WSNs.

In Chapter 3, we propose a novel localized approach to EA design, named localized evolutionary algorithm (LEA), for solving optimization problems. We apply LEA to solve the sleeping schedule and coverage problems in WSNs.

In Chapter 4, we investigate the coverage-preserving routing problem (CPRP) in WSN, and that CPRP is proved NP-hard. Two coverage-preserving routing algorithms, centralized and distributed, are developed to solve the CPRP.

In Chapter 5, we research on the data aggregation problem for object tracking in WSNs. We first propose a localized selection scheme for finding the next node to be

woken up by the active node. Then based on the nodes selected, we propose an algorithm to construct of data aggregation tree.

Finally, Chapter 6 concludes the thesis and discusses our future works.

Chapter 2. Background and Literature Review

In this chapter, we provide a literature review and background knowledge in object tracking in WSNs. The organization of this chapter is as follows. Firstly, Section 2.1 presents an overview of WSNs. Section 2.2 presents an overview of object tracking in WSNs. In section 2.2, subsection 2.2.1 describes the classification of the existing work on object tracking. Subsection 2.2.2 analyses the challenges in designing the algorithm and system of object tracking. Subsection 2.2.3 presents a framework which is able to incorporate existing object tracking algorithms. Finally, subsection 2.2.4 surveys a few representative object tracking systems and summarize the design rules of system.

2.1. Wireless Sensor Networks

Wireless sensor network is a self-organizing network which consists of numerous sensor nodes. The sensor node can sense the data, process the information, and communicate with each other. WSNs are used in a vast variety of fields, including danger detection [HB07], area monitoring [WJ05] [LC11], managing inventory [AS02], and attack detection [HK09]. The basic process of these applications is shown in Figure 2.1. For monitoring the environment, sensors are deployed in a given area to collect and transmit sensing data to the sink.

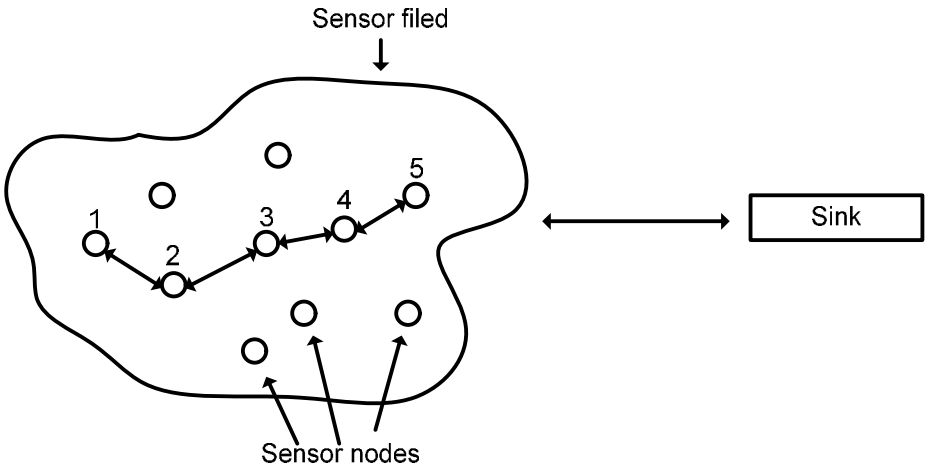

Figure 2.1: Sensor nodes deployed in a sensor field [AS02]

2.2. Object Tracking in Wireless Sensor Networks

2.2.1. Classification of Existing Work in Object Tracking

Based on different assumptions, the algorithm and system will be different. According to three main design assumptions, i.e., the motion mode of object, the number of objects, and network architecture, the existing work on object tracking can be summarized by a three-dimensional classification description. As shown in Figure 2.2, each assumption represents one dimension with a constraint spectrum. The class of the existing work can be obtained by combining parameters from different dimension in various ways. As a result, the taxonomy of existing work (including algorithms and systems) can be yielded. A string of the format X-Y-Z expresses a class in which X represents the employed network architecture (hybrid sensor network, heterogeneous sensor network, or flat sensor network), Y stands for the number of objects to be tracked in the system (multi-target tracking or single target tracking), and Z represents the motion mode of target utilized (predefined motion mode or random motion mode).

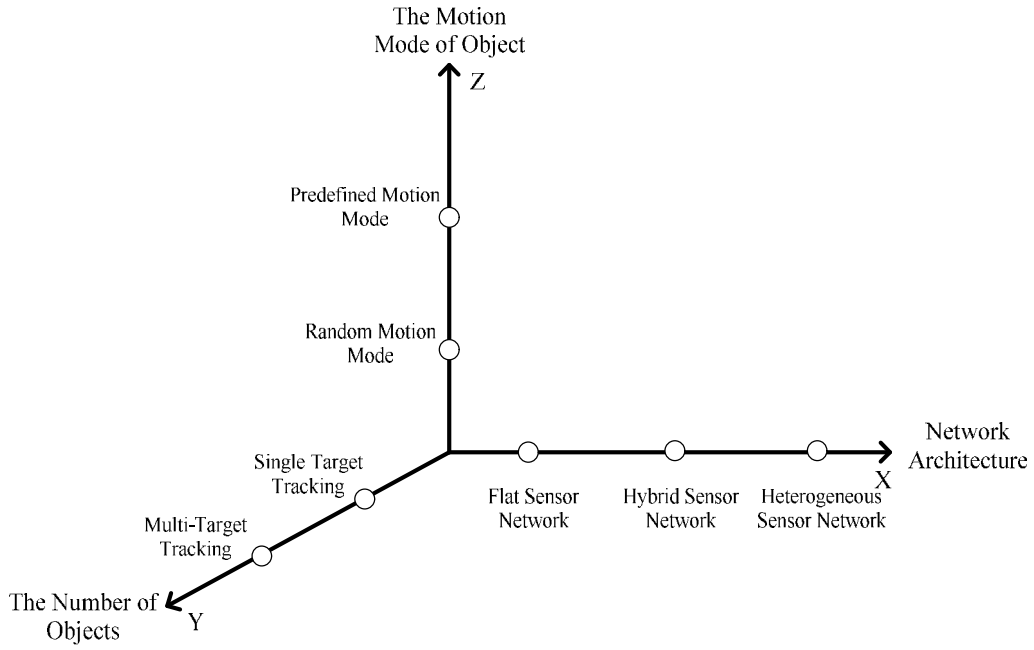

Figure 2.2: 3-dimentional of existing works on object tracking

2.2.1.1. The Architecture of the Wireless Sensor Network

The network architecture can be classified as three classes.

(1) Hybrid sensor network. This network is divided into different layers. Sensors with different capability are distributed in different layers to finish various tasks such as sensing, sending messages, collecting messages, and computing [WZ09] [ZC04].

(2) Heterogeneous sensor network. This network consists of different kinds of sensors, e.g. video, acoustic, light, and others. These sensors work cooperatively to track the target [KA08].

(3) Flat sensor network. Obviously, the sensor in the network has the same function and capability without any difference [ZZ09].

2.2.1.2. The Number of Targets

(1) Single object tracking. A single target goes through a given area [CS11]. When the target is moving into the area, the state of target such as spatial trajectories needs to be estimated by the measurements of sensors. Each sensor node provides a local measurement. However, in most cases, only a relatively small subset of sensors contributes significantly to the estimation, due to the sensing range limitations [RE07] [MS10]. **(2) Multi-target tracking.** Multi-target tracking is more difficult than single target tracking. Firstly, the states of objects need to be tracked simultaneously. Secondly, when the trajectories of the targets overlap, the state of each target needs to be split and identified. Thirdly, when there is no prior knowledge about the number of targets, the number of targets needs to be determined.

For solving first problem, the traditional method is partition of the sensor network. The network is partitioned into equal groups and each group is assigned to track one target. After splitting/merging the groups of sensors, if there is a new target or a target disappears, the state of the targets can be tracked simultaneously. Network partitioning is the common method that widely used to solve the first problem. In [HS09], a seed growing graph partition (SGGP) algorithm is proposed to decide how to split and merge the group to track new targets. In SGGP, the mobile node that is closest to the new target is regarded as a seed node. The seed node initiates a new group and adds its sons into the group by broadcasting. The adding operation stops until the number of sensors in the new group is equal to a predefined threshold. In [Z10], X.Zhang proposes a scheme to partition the network into clusters using the maximum-entropy based clustering criteria. Each cluster corresponds with a target and the cluster head is selected using decentralized particle swarm optimization (DPSO). In [WZ09], a scheme is proposed to select the cluster head and cluster member using a Particle Cardinality Balanced Multi-Bernoulli (CBMeMBer)

Filtering algorithm. To solve the second problem [TS11], Joint Probabilistic Data Association Filter (JPDA)[FS80] is a well-recognized method. JPDA is a single scan filter where the states of existing targets are to be updated based on the weighted combinations of all latest measurements [LC07]. In [TR09], M. A. Tinati et al further enhanced the JPDA filter by developing a distributed JPDA. In the method, the marginal posterior densities of the individual targets are evaluated by using the Monte Carlo simulations and the local GMM parameters for these distributions are calculated by each sensor. Then the average consensus filtering technique is used to obtain the global GMM parameters in each node. The consecutive time steps that the global approximations for the individual posterior densities are obtained from each node are repeated. The disadvantage of JPDA is the number of targets should be known as a prior knowledge. To solve the third problem, in [FZ02], for each distinct target, a corresponding sensor leader is elected. As targets move, new leaders are elected to reflect network changes. The number of targets can be determined by the number of leaders elected. F.Zhao and Guibas describe a distributed mechanism for counting the number of targets in a given field in [FZ03] by using particle and clustering algorithm.

2.2.1.3. The Motion Mode of Target

The motion mode of target can be classified as two classes.

(1) Random motion mode. In this class, the target goes through the area randomly. The trajectory of target is estimated by the sensed data from sensor and previous sensed data [TS08].

(2) Predefined motion mode. The object goes through the area based on a predefined motion mode. The CT (constant turn) and the CV (constant velocity) mobility model are the usual mobility models that are used in object tracking [TB09]. The uniform

acceleration motion (CA) is used in [TB09]. The motion model can also be a directional motion model [KA08] that movement models take into account the moving direction to imitate the actual object motion.

2.2.2. Challenges

The first challenge in object tracking associates with real-time constraint. Real-time problem means the objective of the problem must subject to a strict time constraint. In object tracking, real-time means the state of the target must been identified in short/predefined time delay. In [HV07], the delay of object tracking can be divided into six parts: $T_{initial}$ (the time for initialization), $T_{detection}$ (the time for detection), T_{wakeup} (the time for waking up), $T_{aggregation}$ (the time for data aggregation), T_{e2e} (the time for routing), and T_{base} (the time for computation time on base station). The target tracking system tries to shorten the delay of the parts. However, if the location of the target needs to be estimated accurately, the sensors need collect enough data and use the complex algorithm to compute the result. When the targets enter the area, how to track the target accurately in real-time is a challenging task.

The second challenge is energy efficiency. Because sensor nodes have limited energy, how to extend the lifetime of the sensor nodes is a hot topic of research [AS02]. This problem is usually related to the consideration of tracking errors and tracking algorithms. For reducing the tracking errors, a large number of data and exact algorithm are needed. However, it will consume energy to collect the data and compute the location. Hence, how to improve accuracy and save energy is the main objective of object tracking.

The third challenge is to realize the tracking algorithm in distributed way. Whenever possible, object tracking should be performed in a distributed (in-network) fashion to satisfy real-time constraints and energy efficiency [CH03]. However, the limitations

of the distributed implementation of object tracking should be considered. As to the accuracy of the tracking result, centralized tracking has global knowledge that all measured data is available as the history of the trajectory, whereas distributed tracking is only based on the current limited estimations. The accuracy of the distributed algorithm will be less than the centralized algorithm.

2.2.3. Methods, Techniques, and Algorithms

Since the sensor has limited energy supply, the deployment and operations of nodes are designed with the need to save energy and guarantee accuracy of the result. Based on the characteristics of a complete object tracking process, algorithms used in object tracking are classified according to the stages of object tracking – the deployment stage, the detection stage, and the tracking stage, as shown in Figure 2.3. The focus of the research in different stages is shown in Figure 2.4, Figure 2.5, and Figure 2.6, respectively.

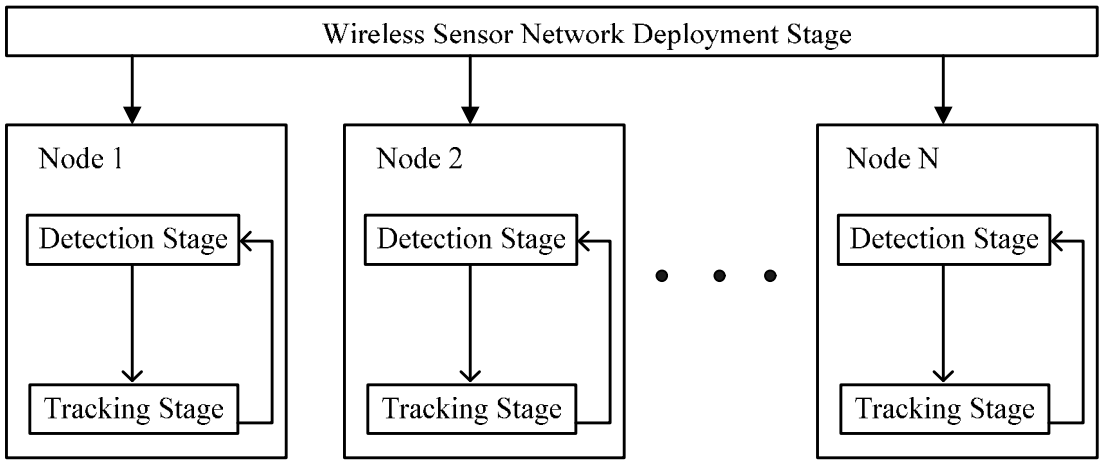

Figure 2.3:The process of object tracking

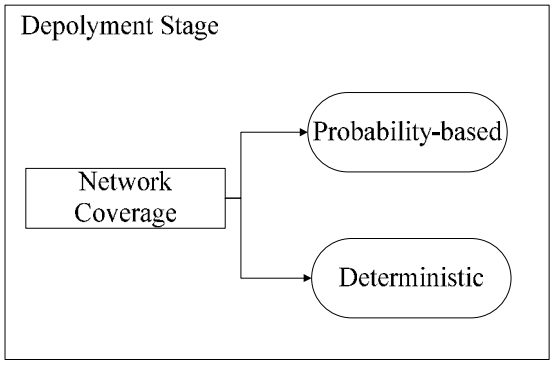

Figure 2.4: The research focus in deployment stage

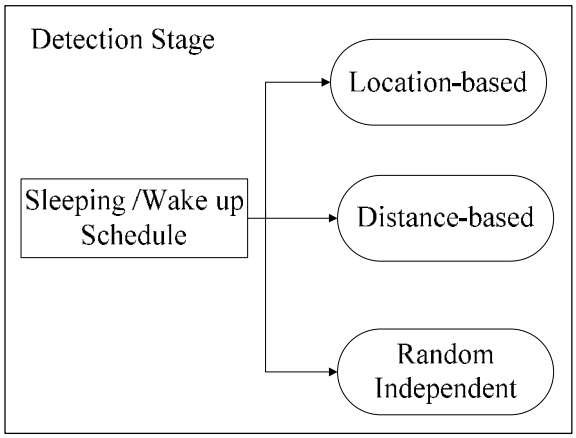

Figure 2.5: The research focus in detection stage

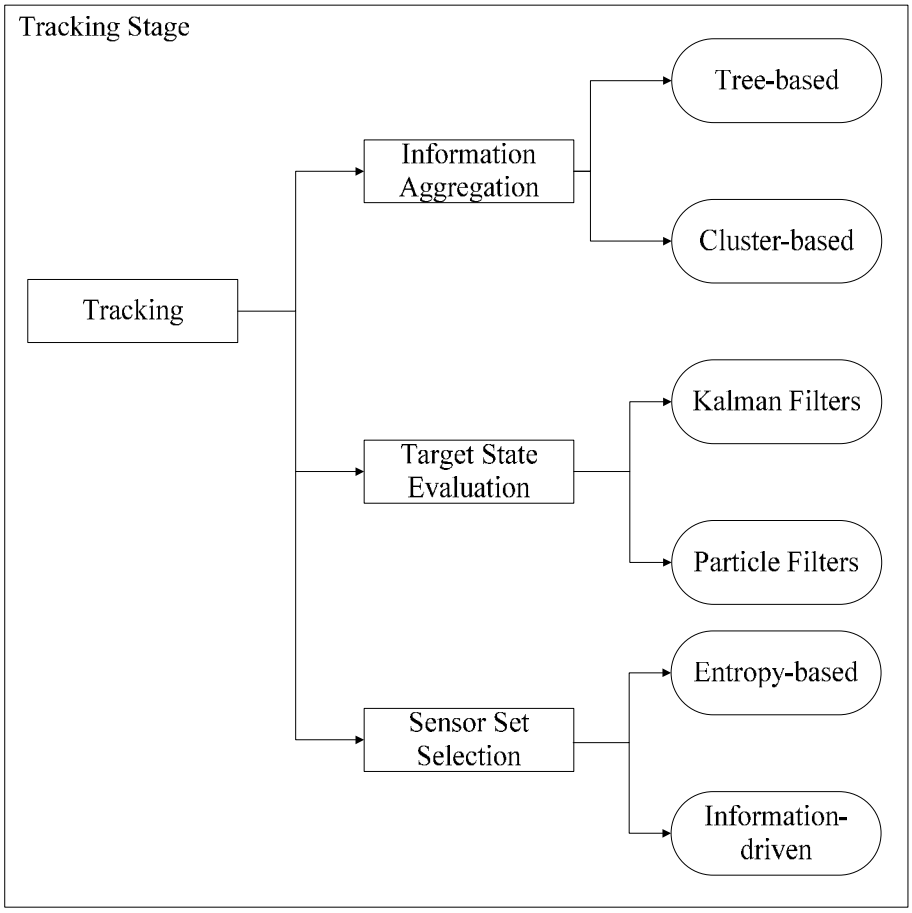

Figure 2.6: The research focus in tracking stage

In the deployment stage, a proper deployment of sensor nodes not only reduces the number of sensors being used, but also enhances the reliability of the network [AS02]. The reliability of the network is reflected by the coverage of the sensing range of sensors. If the sensors can cover the area with the high probability, the detection error will be reduced. The deployment of sensor nodes can be based on deterministic or probability sensing model. However, if the deployment of sensor nodes cannot be controlled, e.g., randomly dropping sensors from a plane to the observed area [M10], the sensor nodes are scheduled to be in different states. Even in the controlled deployment of nodes, a sleeping schedule is also an effective way of alternating sensors between the sleep and active modes in order to reduce energy consumption while maintaining full coverage of the targets [WH08]. Therefore, in

detection stage, sleep scheduling is the main task. Classic methods for node scheduling can be divided into location-based, distance-based, and random independent methods.

When an object enters the WSNs, the corresponding sensor nodes are awoken to the tracking stage. At the same time, other un-related nodes are still at the detection stage. In the tracking stage, information aggregation is a commonly used method. Tree-based and cluster-based aggregations are the two basic forms for gathering sensing and delivering data. Evaluating the state of the target is also an important part of object tracking. Predicting the next movement of the target is very meaningful for studies of detection and save the energy of node.

2.2.3.1. Deployment Stage in Object Tracking

The deployment stage is the initial stage of the whole process of target tracking. In the deployment algorithm, all sensors are deployed by using prior or partial knowledge of the environment, such as the position of the node, the relative distance between two nodes. How to determine the minimum number of sensors that need to be deployed to achieve the optimal level of sensing and communication coverage is the key problem [CT05]. Based on the techniques used for determining the position of sensors [YF06], deployment methods can be divided into deployment based on probability sensing mode and deployment based on deterministic sensing mode.

The deterministic deployment algorithm only depends on the sensing range and distance without considering any probabilistic factors, whereas the probabilistic deployment algorithm is for calculating the better positioning of nodes using detection probability. Table 2.1 shows the comparison between these two algorithms.

Table 2.1: The comparison between deterministic deployment and probabilistic deployment

Technique	Deterministic	Probabilistic
Assumptions	If the object is in the sensing range, it can be detected.	The sensing accuracy of a sensor decreases as the distance to the target increases or the energy of the sensor decreases.
Basic idea	Use the minimum cost/number of nodes to cover the observed area/points.	Calculate the optimal deployment based on the probability of detection.
Advantages	1. It is suitable for the binary sensor network that is usually used in object tracking. 2. It is easy to design the algorithm when the sensors have the united sensing coverage.	1.It is practical in a real world. 2.It considers the detection error and detection accuracy.
Limitations	1. The assumption is ideal. 2. It does not consider detection errors.	1. The detection probability model is only based on the estimation model. 2. Since the sensing range of the sensor becomes irregular when considering with the probability model, the process of calculation becomes complex.

1) Deterministic Deployment Algorithms

Researchers assume that sensor nodes in WSNs can always successfully detect a target within its sensing range, but cannot detect a target beyond the range [ZY06]. The aim with deterministic deployment techniques is to use the minimum number of nodes to cover the observed area so that any target in the area is included in the sensing range of at least one sensor. The output of the algorithms usually is the locations of the sensors. A common method is to dividing the given area into grids/cells [R10]. Then the sensors are assigned into the sets of grids/cells to cover the subareas. Chakrabarty et al. [CI02] applied a grid-based representation to the sensor field and transformed object tracking into the problem of locating a target at a

grid point. In [CI02], the objective of the deployment problem is to determine sensor placement while minimizing the total cost of sensors for complete coverage of the sensor field. Integer linear programming (ILP) method and divide-and-conquer method are used to find the optimal deployment. In [HS04], the objective is to maximize the coverage of the service area while not exceeding a given budget. Evolutionary approach is used to solve the problem.

Different from static deployment, where the positions of the sensors are fixed, a mobile sensor deployment method was proposed in [ST05]. Besides using a relatively large number of static sensors, mobile sensors are used to fill in holes, which are not covered by the static sensors. Using mobile sensors can greatly reduce the number of static sensors needed to achieve full coverage. However, it is possible that a target appears within a hole that no mobile sensors can detect. Therefore, how to apply different optimization methods to optimize the deployment is still a challenging issue.

2) Probabilistic Deployment Algorithms

Although deterministic deployment is simple for analysis, in reality the detection of a target is not a deterministic process [ZY06]. A target is not always detected by a sensor because the sensing accuracy of a sensor decreases as the distance to the target increases or the energy of the sensor decreases. Since probability-based deployment algorithms in a real environment are more practical than in a testing environment, they have become the main focus in studies on the coverage of sensors.

Zhang et al. [ZY06] proposed a differentiated node deployment algorithm called DIFF_DEPLOY, which is based on a probabilistic detection model. Not only was the miss detection probability considered, the detection requirements for different

locations also were included in the optimization problem. The algorithm could be used in both precise node deployment and node deployment with uncertainty.

Inspired by the combination of attractive and repulsive forces in physics, Zou and Chakrabarty [ZC+07] applied a virtual force algorithm (VFA) to enhance the coverage of sensors after their initial random placement. The resulting probabilistic target localization algorithm is executed by the cluster head in a cluster-based sensor network architecture. The cluster head calculates the percentage of its local area coverage based on a probabilistic node detection model. The factor of the distance between the target and the sensor was considered and moving instructions were sent to redeploy the sensors.

Dhillon and Chakrabarty [DC03] proposed two algorithms for placing sensors in a sensor field. They assumed that the probability of detecting a target by different sensors were different. The probability was defined as a function whose value decreased exponentially with the distance between the target and the sensor. One of algorithms maximized the average coverage of the grid points in the field. The other algorithm maximized the coverage of the most vulnerable grid points. The two algorithms cooperated to find an efficient node deployment under the constraints of imprecise detections and terrain properties.

3) Metrics

As to deployment algorithms, the target is to deploy fewer sensors to achieve a greater range of coverage. The number of sensors that should be placed is the most popular performance metrics used in deployment algorithms. It is usually compared under the same predefined miss probability threshold [ZY06] or the same predefined sensing range. If more than one kind of sensor needs to be placed, the density of the

sensors and the total monetary cost of the sensors are also performance metrics for the deployment algorithm.

4) Open Issues

Though the deployment of the sensor has been studies extensively, there are still several interesting issues deserve further investigation. The first one is the deployment of sensors in a three dimensional WSN. Most of the existing works focus on the deployment problem in a 2-D WSN. However, in some applications, such as tracking the pollution in sea [WR10], deploying 3-D WSN is needed. There are a few work based on the problem, for example, in [HT04], the authors reduce the problem from a 3-D space to a 2-D space, and further to a 1-D space to find the optimal deployment. In [MS$^+$10], the paper proposes a new heuristic algorithm for computing a near optimal sensor node deployment that minimizes the cost for achieving the full coverage and node connectivity of a 3-D target space with obstacles. Solid space theory, division of the space, and reduction of the space are the possible solution for the problem. The second open issue is deployment in the heterogeneous network. Since the network consists of different kinds of sensors with different sensing model or detection model, how to find an optimal deployment is a difficult task [WC07]. In [WK04], impact of heterogeneous deployment on lifetime sensing coverage in sensor networks is discussed in single-hop and multi-hop communication.

2.2.3.2. Detection Stage in Object Tracking

1) Sleep Scheduling

Generally, sensors are excessively deployed in the observed field. A portion of the sensors is able to give complete coverage of the area. Therefore, some redundant

sensors can enter a sleep mode to save energy. When needed, a set of sleeping sensors are woken up to full activation [WH08]. The classic sleep scheduling techniques can be classified as “location-based”, “distance-based”, and “random independent”. Table 2.2shows a comparison of these algorithms.

(1) Location-based Sleep Scheduling. The location-based sleep scheduling requires each node to know its accurate location through a global positioning system (GPS) or location service. Initially, all nodes are in the active state and then the redundant nodes are found using location information of node. The non-redundant nodes remain active, while the redundant nodes change into the sleeping mode.

Wang et al. [WX03] proposed a coverage configuration protocol (CCP) that can provide different degrees of coverage under request. In a CCP, each node stays in one of the three states: SLEEP, LISTEN, and ACTIVE. HELLO, JOIN, and WITHDRAW messages are used to reduce contention among neighbours in the transition from LISTEN to ACTIVE and the transition from ACTIVE to SLEEP, respectively. Under the assumption that the communication range with other sensors is less than twice the sensing range, CPP is integrated with Span [CJ02] to form a connected covering set.

Zhang and Hou [WH05] proposed an optimal geographical density control (OGDC) algorithm to schedule the sleep and wake up times based on a triangle tessellation process. The objective is to maintain sensing coverage and connectivity using a minimum number of active sensor nodes. Each node can be in one of the following three states: UNDECIDED, ON, and OFF. Each round is composed of a node selection phase and a steady state phase. At the beginning of the node selection phase, every node is powered on with the UNDECIDED state. After the remaining energy-based selection is performed, the starting active node broadcasts its information. The node that is not redundant and has the minimum deviation to the

desired direction and angle required by the already active nodes will have the largest chance to become active next. After each node has determined its state, the process will terminate.

Table 2.2: The comparison among location-based, distance-based, and random independent techniques

Technique	Location-based Sleep Scheduling	Distance-based Sleep Scheduling	Random Independent Sleep Scheduling
Assumption	Each node is aware of accurate location.	Physical distance-based: The distance between the nodes is pre-knowledge or the distance between the nodes can be calculated. Hop distance-based: The hop count between nodes is pre-knowledge.	The time is divided into cycles based on a time synchronization method
Basic idea	Each node has predefined state at first and then the redundant nodes are found by using location information. The non-redundant nodes remain active, while the redundant nodes change into the sleeping state or other states.	Physical distance-based: The distance is calculated by the number of neighboring nodes, and then the redundant nodes are obtained by using geometry method. Hop distance-based sleep scheduling: Each node maintains the hop table. The state of node is determined by the hops.	Each sensor enter the sleep state randomly by using messages and the decisions of sleeping/wake up are made independently based on the probability mode.
Advantages	1. The redundant nodes that are calculated accurately because GPS/location service accurately determines their location. 2. Without the complex computing, the redundant nodes can be determined quickly.	1. It doesn't need accurate locations which is practical in real world. 2. The algorithm can only use the local and simple information to find the redundant node.	1. It is suitable for distributed /local computing. 2. It is energy-efficient and light-weight because each sensor doesn't require any interaction with their neighbor.
Limitations	It needs to know the geographical information of sensor or the localization information, which are considered expensive and infeasible to obtain.	The process of calculation is time-consuming if using the geometric method to calculate the coverage area accurately.	It is not robust against unexpected failures that destroy the sensors before they run out of energy.

The CCP and the OGDC are both distributed algorithms. The OGDC is fully localized because its scheduling decision is made based on one hop neighbourhood information. The authors of [WH05] also compared the performance of these two algorithms. The paper showed that the performance of the CCP declined as the number of deployed nodes increased. The OGDC took less time to configure the sensor network than the CCP. A smaller number of working nodes was needed by using the OGDC and the resulting sensor network by the OGDC had a longer lifetime than the CCP.

(2) Distance-based Sleep Scheduling. Because the location-based sleeping schedule algorithms need to know the geographical information of the sensors or the localization information, which are considered expensive and infeasible to obtain, distance-based sleeping scheduling algorithms are proposed. The distance information between nodes can be the concrete distance calculated by the geometric method or measured by the received signal strength based on an attenuation radio transmission model [YK07] or be the hop counts used for transmitting information between any two sensors [WH08] [WW06]. According to the distance information, geometry method is used to find the redundant nodes.

In the physical distance-based Sleep Scheduling, Younis et al. [YK07] proposed a location-unaware coverage algorithm (LUC), which selects nodes to be active while the other nodes are put to sleep. In this paper, the four redundancy check tests are applied by using the distance between two-hop neighbours and the number of neighbouring nodes. In the first and second redundancy check tests, the coordinate of the node needed to be checked is assumed as (0,0). Then the coordinates of the neighbouring nodes are assigned according to the distance between the node checked and its neighbour. Based on these coordinates, if the node checked is in the triangle in which the vertexes are the coordinates of any three neighbouring nodes, the node checked is redundant. In the third and fourth redundancy check tests, based on the

uniformly deployment, if a node V has more than four active neighbours within the sensing range R_s of the node V, and if every neighbour of V is also a neighbour of these active neighbouring nodes, V is regarded as a redundant node.

In [ZC+07], the distance between two nodes is estimated by the number of neighbouring nodes. Then the area is divided into non-overlapping triangles, and the vertices of these triangles are the active nodes. The basic idea for selecting an additional node to be active is that the ratio of the size of the vacancy or white area inside the triangle to the area of the triangle should be less than or equal to a predefined threshold.

In the hop count-based Sleep Scheduling, Wang et al. [WF07] proposed a layered diffusion-based coverage control (LDCC) protocol, which exploits the hop count information to select active sensor nodes. In this paper, the hop counts are the transmission times for delivering a packet from a sensor to the base station. Wang et al. [WW06] proposed a hop-based sleep scheduling algorithm (HSS) for enhancing the lifetime of a WSN. According to the average hop distance to the sink, the HSS divides the sensor network into several levels. The probability of putting a node to sleep is related to the level and the density of the sensors. Sensors that are farther away from the sink node have a greater probability of being in the sleep mode. As the HSS algorithm needs to know the information of all levels, it is a centralized algorithm.

(3)Random Independent Sleep Scheduling. Random independent sleeping (RIS) algorithms [GM04] [KL04] belong to self-scheduling algorithms, which make each sensor enter the sleep state randomly and independently of each other. An RIS algorithm has the following major characteristics according to [WX06]: (1) it does not require location or distance information; (2) nodes do not maintain a neighbourhood table; (3) because the sensors do not dynamically evaluate their

situation, the basic RIS mechanism is not robust against unexpected failures that destroy the sensors before they run out of energy.

The probing environment and adaptive sleeping (PEAS), proposed by Ye et al. [YZ03], is a self-scheduling method. It consists of a probing environment algorithm and an adaptive sleeping algorithm. In the probing environment mechanism, all nodes are sleeping for an exponentially distributed random time. When the time passes, the node wakes up and sends a PROBE message within a certain range. If it does not receive any reply, it wakes up and starts working. In the adaptive sleeping mechanism, the aggregate probing rate is calculated by each working sensor based on its sleeping neighbours. The measured probing rate is then included in a REPLY message from the working node to its probing neighbours. The probing nodes adjust their sleeping times according to the probing rate in the message.

2) Metrics

Because the design goals of the sleeping schedule are sensing coverage, connectivity coverage, and energy cost, the performance metrics of sleeping schedule include the energy consumption or lifetime of the network, the number of working nodes required, the average coverage ratio (the ratio of the covered area to the total area to be monitored), and the latency (the delay from the time that the target shows up to the time that the target is detected).

3) Open Issues

How to design an efficient sleep scheduling method is still an open research question. Most of the papers on problems of coverage are based on a unit disk-sensing range in WSN. However, in practical environment, sensors have irregular shapes of sensing area. For example, as shown in Figure 2.7, cameras and sensors

with antennas, the sensing area is a sector. However, if considering the irregular shapes of sensing ranges, there are three difficulties. The first is how to adjust the angle of each node to cover the area. The second is how to calculate coverage area based on the irregular shape of the sensing range of sensors in a given region. The third is how to schedule the nodes and find the redundant nodes if the angle of the sensor is adjustable.

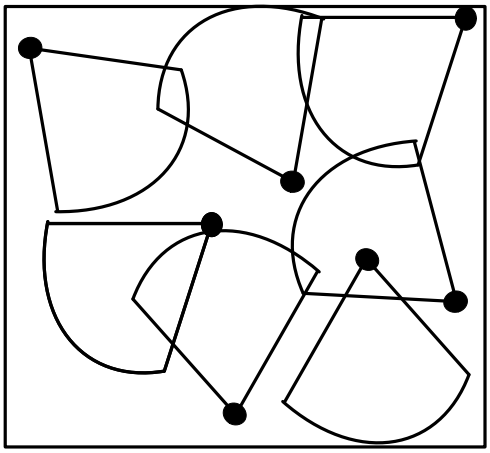

Figure 2.7: The sensor with sector sensing area

2.2.3.3. Tracking Stage in Object Tracking

When a target is detected by one or more sensors, the sensors around the target are woken up to detect collaboratively. When the target is moving, the sensors should always track its information continuously. Using the minimum number of sensors to get high tracking accuracy is the objective in this stage. There are three general phases in the tracking stage: information aggregation, the target state evaluation, and selection of next sensor set.

1) Information Aggregation

The process of information aggregation can be as follows. The sensor nodes surrounding the moving target should promptly provide robust and reliable status information about the mobile target and the region around it in an energy-efficient way. The network should also forward this information to the sinks in a fast and energy-efficient way. The main objective of the information aggregation is to minimize the amount of communication so as to increase the lifetime of the network. The algorithms of information aggregation usually include two types: tree-based and cluster-based. Table 2.3 shows a comparison between these algorithms.

(1)Tree-based algorithms. As in Figure 2.8: An example of a data aggregation tree, there are k sources $S_1, S_2, \ldots, S_k$ and a sink D. Suppose the graph $G=(V,E)$ consists of a set of nodes V and a set of edges E. Building a data aggregation tree is that finding efficient paths from the k sources to the sink D through a set of intermediate nodes that allow in-network consolidation of redundant data.

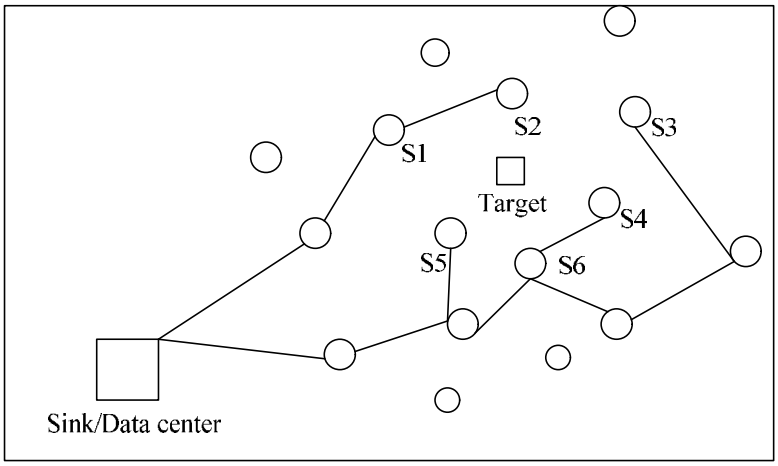

Figure 2.8: An example of a data aggregation tree

Table 2.3: The comparison between tree-based and cluster-based techniques

Technique	Tree-based	Cluster-based
Assumption	There are source nodes and sink nodes. The source node can connect with sink node through multi-hops.	The sensors in the cluster can connect with the cluster head. The cluster head can connect with the sink node.
Basic idea	The tree is commonly composed of a sink, intermediate nodes, and nodes that detect the target. The sink is viewed as the root; the nodes that detect the target are viewed as leaves. When the target is moving, the tree is constructed dynamically according to the resource usage, tracking quality.	The network is divided into clusters. The cluster heads are elected by the election algorithms. The information is detected by cluster members and then is delivered to the cluster heads.
Advantages	1. In the tree topology, the sensor only maintains the parents' and children's information. 2. The information from source nodes can be combined/aggregate at the middle levels which save the energy and message. 3. The evaluation metrics can be mapped to the weight of the edge.	1. The network becomes a hierarchical structure, which is beneficial for applying the different algorithms to the different levels of the network. 2. The sensors can only communicate with other sensor in the same cluster which can be easily utilized to design the distributed algorithm.
Limitations	The algorithms are usually centralized, since global information is needed when the tree is constructed.	The cluster head needs to manage the members in the cluster which consumes much energy. Thus, the energy of the cluster head will be used up quickly. The energy balance of the network is broke. Furthermore, the features of cluster make the cluster-based algorithm is not suitable in the flat wireless sensor network.

Zhang and Cao [ZC04][ZC^{+}04] proposed a dynamic convoy tree-based collaboration (DCTC) framework to detect and track a mobile target and monitor its surrounding area. The schemes divide a sensor network into grids, with a grid head in each grid. Only the grid head is awake when there is no target near the grid. When a target enters the detection region, the corresponding grid heads wake up more sensors. These sensors collect sensing information and cooperate to elect a root and construct

a convoy tree. The root collects and refines the information from the other sensors about the target by using some classification algorithms. Some active sensors change to sleep when the target leaves its sensing range, while some other sensors are activated by the root for continuous tracking. Therefore, the convoy tree and the root are dynamically altered and changed to optimize the communication overhead.

In [HI04], a dynamic sink-oriented tree is constructed. The nodes that are only one hop away from the sink become root nodes, and are responsible for sending information on the connection or disconnection status of the mobile sink to its descendant nodes. If the sink moves, some root nodes will become disconnected from the sink. As soon as the timer has expired, the root nodes flood a JoinREQ message to their neighbours. The nodes receiving the JoinREQ make the sending node as their descendant by corresponding with a JoinREP message.

Lin et al. [LP06] developed several tree structures for in-network object tracking, which took the physical topology of the sensor network into consideration. The proposed method is a two-stage approach, with the aim being to reduce the update cost and the query cost, respectively. In the first stage, an object tracking tree is generated. The authors discussed two solutions to the object tracking tree: a deviation-avoidance tree (DAT) and a one-based DAT (Z-DAT). In the second stage, a query cost reduction (QCR) algorithm is developed to adjust the tree in the first stage for a further cost reduction.

The drain-and-balance (DAB) method proposed in [KV03] constructs the tree in a bottom-up way. Within each DAB step, a subset of the sensors is merged by utilizing the event rate information. Sensors are partitioned using one or more event rate thresholds, and the high-rate subsets are merged first.

(2) Cluster-based Algorithms. Chen et al. [CH03] proposed a clustering algorithm for target tracking using acoustic information. They regarded the network as a hierarchical structure, made up of a static backbone of high-capability sensors as cluster heads (CHs) and a low-end sensor network for providing the sensing information to the CHs. By using the Voronoi diagram which means a special kind of decomposition of a metric space determined by distances to a specified discrete set of objects in the space, dynamic clustering was realized, ensuring energy conservation and sufficient information to determine the location of the target.

Medeiros et al. [MP08] proposed a distributed object tracking system that employed a cluster-based Kalman filter in a network of wireless cameras. The cameras can observe the same target and cooperate with each other to form a cluster and elect a cluster head. The advantage of using such a cluster-based method is that the uncertainty of the position of the moving target can be reduced, making the localization more accurate. The experimental results show that the proposed tracking protocol requires much fewer transmission messages than a centralized tracker, while achieving comparable tracking accuracy.

Some other methods also combine cluster with tree topology to solve the problem of aggregation. For example, Fang et al. [FZ03] proposed a distributed aggregate management (DAM) protocol, in which nodes that detect energy peaks become cluster heads, and a tree of cluster members is formed by neighbours that detect lower energy levels.

(3) Open Issues. Since the target is moving continuously, the nodes that can sense the target will be different from time to time. Therefore, the first open issue is how to dynamically select the minimum number of nodes involved to deliver the sensed data to the sink. Though there are many paper worked on this problem, most of them focus on tracking a single target. In multi-target tracking, it needs to be considered

that combining, reconfiguration, and division the tree/cluster with minimum number of involved nodes in designing the aggregation algorithm. The second issue is how to construct the desired cluster/tree as soon as possible and how to evaluate the delay. The researchers are still finding an algorithm which can deliver the information to the data centre quickly and accurately. In tracking the pollution in the ocean, there is an interesting problem. When the wave is waked upon the sea, the topology of the sensor network will be changed. How to construct dynamically the cluster/tree in real time in the dynamic topology is a great challenge.

2) Target State Evaluation

When the information from sensors has been gathered by the data centre/sink, the data centre/sink needs to associate the measurements to calculate the state of target, the state can be the position or attributes of the target. However, only set of measurements contribute to the final results. Some measurements from sensor are not accurate which are regarded as noise. Kalman Filters (KFs) [WB01] and Particle Filters (PFs) [GS93] are the representative methods for evaluating the state of target and filter out the noise.

(1)Kalman Filter. KF [AB11] is a set of mathematical equations that provides an efficient computational solution to discrete time data filtering problems, in essence removing extraneous noise from a given stream of data. A KF has two distinct phases: predict and update. The predict phase uses the state (e.g. location) estimate from the previous timestamp to produce an estimate of the state at the current timestamp. In the update phase, measurement information at the current timestamp is used to refine this prediction to arrive at a new, (hopefully) more accurate state estimate, again for the current timestamp. In [BC06], the object tracking system is modelled by using KF to filter out the measurement noise and estimate the path of object. The system consists of four stages: the state space calculations, the

measurement model, prediction stage and correction stage. In [SK11], KF is used to denoise the signals based on polar coordinates.

Besides the KF above, the Distributed KF (DKF) is used for locally estimate the state of target. The aim of a distributed filter [HR10] is to parallelize the state estimation such that every node determines its state estimate independently. Abdelgawad [AB11] et al propose a novel DKF which is based on a fast polynomial filter to accelerate distributed average consensus in static network topologies. The proposed DKF consists of consensus filter and micro KF on each node. In [HR10], a data fusion step is added to the filter. This allows the usage of more measurement information available in the network to improve the accuracy. It can be used in dynamic topology network. Other modified KF, such as the extended Kalman filter (EKF) [SL93] [MA08], the mixture KF (MKF) [DL07] and the switching KF (SKF) [MM05] are also used for object tracking in WSNs.

(2) Particle Filter. Recently, PFs are applied for solving object tracking widely. In PFs, a set of 'particles' represents the candidate state values [IC05]. The filter evaluates how well individual particles correspond to the dynamic model and set of observations, and forms a state estimate through an appropriate weighted averaging of particle values (or perhaps a maximization). It often consists of four steps [CI05]: a propagation step, the update step, the estimation step and the resampling step. Gordon et al. [GS93] developed a bootstrap filter (also named the sequential importance resampling (SIR) PF). The filter introduced a resampling step to prevent the filter from diverging, which removed the particles with the lowest weights at each step and created new particles at points where the weight was the highest. The bootstrap filter was shown to be more accurate than the EKF for tracking in a system with nonlinear measurements, such as bearings-only tracking. Since then, several variants of this bootstrap have been developed, such as the bootstrap for multi-target

tracking [HC02] [VN01] and for manoeuvring targets using an interacting multiple model (IMM) PF approach [MI02].

Since PFs have been shown to be particularly effective in a distributed sensing environment [LC04], a Distributed PF (DPF) is proposed in [C04]. The DPF [IC05] maintains a local particle filter at selected nodes. Through communication, each local filter on the node performs estimation and to implement extensive compression of local measurements for transmission to other nodes. In [OB08],a decentralised data fusion architecture with channel filters is proposed. Local PFs update observation likelihoods from local sensors and fuse information received from channel filters. Channel filters maintain a record of common information between two nodes. Garrick Ing et al propose two improvements to DPF to reduce communication overhead. The first improvement step is that performs Huffman coding by constructing the tree from the information of PF. The second improvement is that reduces the fraction of communication energy wasted through transmitting packet headers. A thorough description of different types of PF can be found in [AM02].

(3) Open Issues. Designing a distributed version of the filter is the hot research area. Because of the distributed property of the particle filter, DPFs are studied in recent papers. How to modify DPFs using local information estimation is worth to be investigated. Besides the filters, design of the simple and quick method for status estimation is the open issue in object tracking. For example, in [WB08], the author proposes an algorithm to estimate the target location and the target velocity by using reduction from a two-dimensional area into a one-dimensional arc in a distributed and asynchronous manner. The method is suitable for solving the tracking problem in a densely deployed WSN. For a sparsely deployed WSN, Deepak [JK11]et al proposes an approach that the location of a moving target is approximated by tangent estimations to three circles, each representing range of a sensor. Most of the filters

only consider one type of noise mode. However, when tracking a target in a real work, there exist several types of noise, such as environment noise, measurement noise, and transmission noise. Thus, designing a robust algorithm in the presence of different types of noise is an open issue. In [ZZ09], the paper converts the original tracking problem to the problem of finding the shortest path in a graph, which is equivalent to optimal matching of a series of node sequences. Then, a tracking framework using node sequences is proposed which can be robust to the different types of noise models.

3) Sensor Set Selection

The simple way to guarantee the sensor network tracks the moving target continuously is that all the sensors are waken up and keep active until the target goes out from the network. However, it will consume much energy and the lifetime of the network will be shortened. Therefore, for saving energy, when the target is detected, only a set of sensors around the target needs to be woken up. For tracking the target continuously, the current active sensors wake up a next set of sensors to track the target based on the possible location at next time slot predicted by the sensors/data centre.

There are many existing algorithms to predict a next sensor set. The basic idea of these algorithms is that firstly the next location is predicted according to the current measurements and the motion model of target. Then a set of sensors which are near the predicted location is selected according to the residual energy, required coverage, or the type of information required. Entropy-based solutions where the selection schemes aim to minimize the entropy of measurement is the common method in object tracking algorithms [RE07]. The paper [MS10] proposes a maximum mutual information under energy constraints-based sensor selection approach. The basic idea of the sensor selection scheme has five steps. At first, the sensors in the R_{max}

range are selected. R_{max} is equal to the maximum sensing range. Then, Compute the mutual information function based on the predicted target position. The third step is to select the best sensors by maximizing the mutual information under energy constraint. Quantize the sensors' measurements and execute the quantized variation filtering (QVF) algorithm in the fourth and fifth step. QVF is a method which is utilized to estimate the position of the target and predict the next position.

The information-driven solution is also a common method for set selection. The information can be the resource usage, or the track quality. In [AL08], Loredana et al propose a selection procedure to find informative sensors to minimize the energy consumption of the tracking task. According to the predicted location of the target obtained by PF, a node V_a near the predicted location is woken up. The neighbouring nodes of V_a within a predefined threshold of RSS are grouped. Finally, a subset of nodes N_d (N_d is predefined) is selected to guarantee that the sum of the energy consumption of the sensors in the subset is minimized.

Open Issues. In most of existing work, a node which is nearest to the predicted location will be woken up at first. Then, the node will select other nodes which are within a predefined range. Then the most informative nodes are woken up from these nodes. However, the approach mentioned above is not fully distributed algorithm and consumed time and energy when the selection procedure is executed from one node to many nodes and then to several nodes as shown in Figure 2.9. Therefore, how to perform a selection scheme in fully localized way and in real-time is a challenging issue. For example, if an active node can wake up a next node one to one which is shown in Figure 2.10, the time and message will be saved.

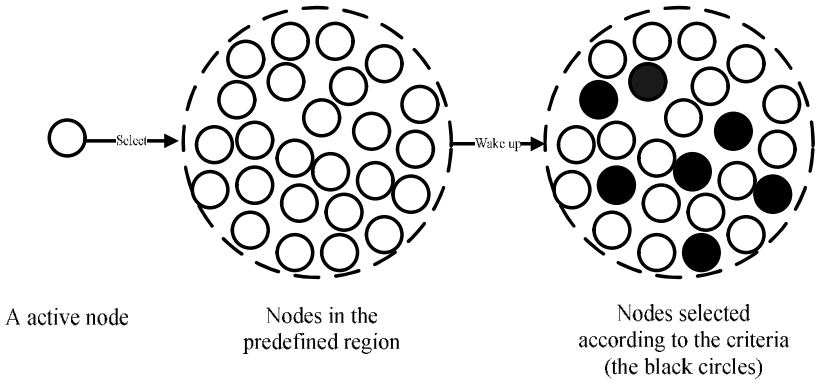

Figure 2.9: A selection process in the existing work

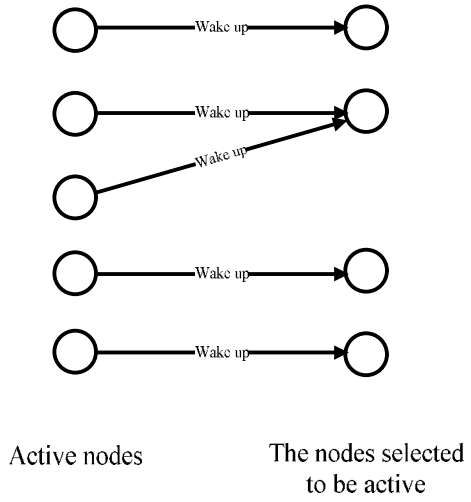

Figure 2.10: A proposed selection process

The other interesting topic in sensor selection is how to recover the error. That means when the selection procedure is executed according to a prediction failure, how to reselect the right nodes based on the current status of target quickly. In [YF06], a mesh approach is proposed to helping the system recover from the prediction failure efficiently. It assume that the speed of target is V_t at the time t, then it is not possible

for the target to move out of the region C called the mesh region within time period $t + t_1$. The sensors which can cover region C completely will be selected to wake up.

4) Metrics

The trade-off in target tracking is between energy efficiency and tracking errors, so the main metrics of this stage are: 1) prediction error: which refers to the difference between the estimated trajectory and the real trajectory; and 2) energy consumption: which refers to the total number of Joules consumed for one whole simulation.

2.2.4. System

Several architectures and models have been proposed for object tracking systems. This section presents some famous and classic systems.

2.2.4.1. VigilNet

VigilNet, proposed by Tian He et al. [HK06], is a popular system for tracking the moving vehicles. The system is composed of four major groups: initialization, tracking, power management, and general utilities. Its structure is depicted in Figure 2.11 and its components are described as follows:

1) Initialization. Initialization components are responsible for the establishment of basic infrastructure. *2) Tracking.* Tracking components support the event tracking functions. The VigilNet tracking operation has six phases: Initial Activation, Initial Target Detection, Wake-up, Group Aggregation, End-to-End Report, and Base Processing. Figure 2.12 shows the tracking operations in VigilNet [HP07]. In tracking, the localization module is responsible for ensuring that each mote is aware of its location. The main contribution of the group management component, described in [BB03], is to establish a unique one-one mapping between a group and

a physical event as well as to add and delete members of the group as the event moves through the environment. The aggregation management is in-network aggregation by organizing the motes into groups. *3) PowerM.* The PowerM module performs power management, which puts motes to sleep as described earlier, when no significant events are detected. *4) General utilities.* Utilities are used for facilitating downloading, debugging, tuning, and statistical logging.

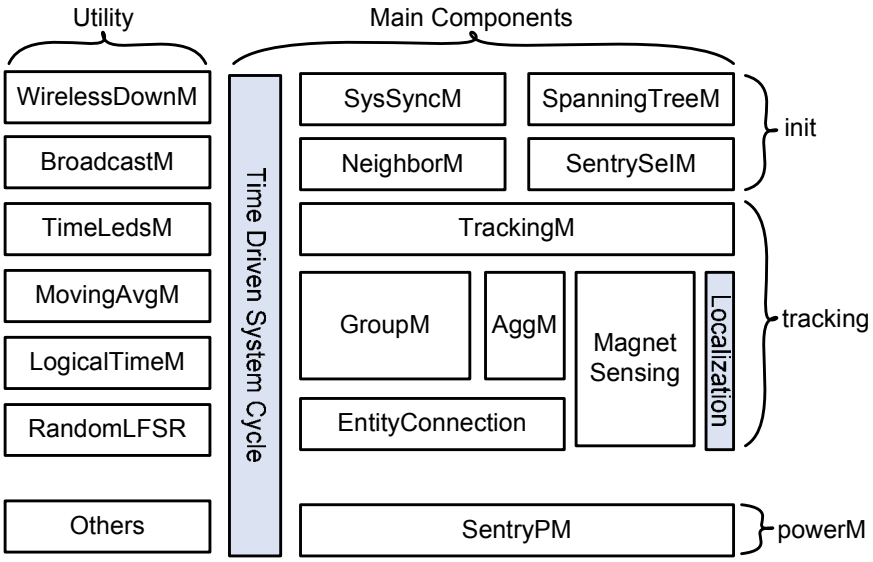

Figure 2.11: VigilNet system architecture [HK06]

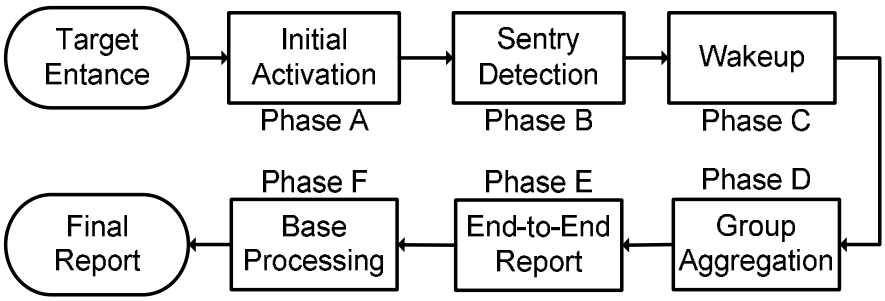

Figure 2.12: Tracking operations in VigilNet [BB03]

The VigilNet was proposed for military research, so its terminology and original application is defence-oriented. The drawback of the VigilNet is that each state needs to be sent to the base station to calculate which will cost more in terms of communication.

2.2.4.2. HTM (Heterogeneous Tracking Model)

The heterogeneous tracking model is a prediction-based object tracking model. The model includes three phases: the Data Collection and Mining Phase, the Prediction Phase, and the Recovery Procedure. In the Data Collection and Mining Phase, a cluster head collects the moving object's message and exploits the variable memory Markov (referred to as VMM) model for mining object moving patterns. Then, the cluster head predicts the next movement of the object. When a cluster head cannot precisely predict the sensor nodes to detect an object, the recovery procedure will be performed to track the object. A recovery procedure will be executed by waking up sensor nodes within the coverage region of the cluster head. Based on HTM, only a bound number of sensor nodes need to participate in the recovery procedure.

In contrast to the VigilNet, the HTM does not provide a systemic view. Instead, it executes the object tracking from the task view and achieves in-network mining in that mining object moving patterns are performed while moving records are forwarded to the sink.

2.2.4.3. MLOT (Multi-Level Object Tracking Strategy)

Figure 2.13 shows the system architecture of MLOT [TL09]. The system workflow consists of three main phases: (1) the clustering of sensor nodes; (2) the discovery of movement patterns; and (3) the prediction and recovery of locations of moving objects. At first, the sensor network is clustered as a multi-level structure by a

clustering mechanism, such as K-means [HK00].Then, the movement logs of the moving objects are analyzed by a data mining algorithm to obtain the movement patterns. Subsequently, the next location for a moving object in sensor networks is predicted and a set of sensor nodes are activated.

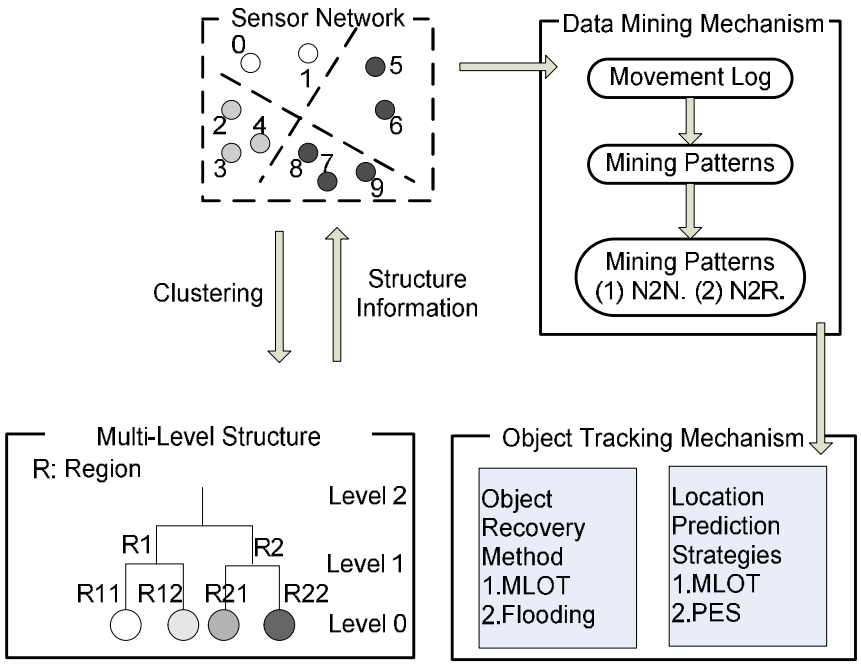

Figure 2.13: System architecture [TL09]

In contrast to the VigilNet, there is no hierarchical system that consists of sensors with different capablity in MLOT. Instead, it pretreats the wireless sensor networks and combines the data mining mechanism with an object tracking mechanism to obtain a more accurate result. Unlike HTM, there are explicit components to the system. The data mining component is responsible for mining movement patterns and the object tracking component includes location, prediction, and recovery steps.

2.2.4.4. Design Principles

Energy efficiency and tracking accuracy are the key objectives of these systems. Based on the above discussion of the systems, the design principles of the object tracking systems can be summarized as follows:

A generic framework of the system should include three layers at least: physical layer, network layer, and application layer. In physical layer, sensors and devices are deployed according to the predefined structure. The network layer is utilized to control the topology and responsible for the communication and routing. In the application layer, different algorithms and functional components are combined to track the target.

Due to the limited energy of sensors, power management component should be designed to monitor the energy consumption when tracking a target. The component should run through the whole the system and control the energy at each phase of the system.

For the real-time requirement in object tracking, the time controller/timer or time synchronization component should be included in the system. The generic system workflow consists of three main phases: (1) collection of the data; (2) discovery of the status of the target (e.g. localization); (3) prediction.

Chapter 3. LEA: A Localized Approach for Evolutionary Algorithms

In this chapter, we describe the proposed localized approach for evolutionary algorithms (EA) for solving the optimization problem in large distributed system. This chapter is organized as follows: Section 3.1 presents related works in EAs. Section 3.2 describes the proposed LEA framework in detail, including the implementation steps in the initialization, local search, coordination, and final result evaluation methods. Section 3.3 discusses the number of search subspaces and evaluation metrics of the LEA. Section 3.4 presents the suitable applications and an example of using the proposed LEA framework to solve a coverage problem in WSNs. The performance of the proposed algorithm and the analysis of the results are also presented. Section 3.5 concludes this chapter.

3.1. Overview

Evolutionary Algorithm is one of the popular approaches for solving optimization problems in various kinds of applications. However, for obtaining the global optimal solution, a processor needs to 1) store the global information of the whole problem which consumes storage; 2) search the candidate results in the global search space which consumes time. In addition, the existing EAs assume that each processor has the prior knowledge of global information. Because collecting the global information is time-consuming when the corresponding information is large and distributed on many processors, the scale of the network will raise undesirable problems, especially in the very great systems.

Because of the comparatively high computational time required to implement EAs, there are many existing works that try to speed up the computation in two aspects, 1) speed up the convergence of the algorithm, 2) search the solution in parallel way. For

the first aspect, Ant colony optimization (ACO) as an extension of EAs is proposed for discrete optimization problems [D04] and particle swarm optimization (PSO) as another extension of EAs is proposed for continuous optimization problems. For the second aspect, the parallel EA (PEA) is also proposed to speed up the search process [OLP03] in parallel way. ACO simulates the behaviour of ants in seeking a path between their nest and food to solve the best path problem in a graph. The central part of the algorithm is a pheromone model which means that the ants make a decision according to the pheromone. The future decision is affected by the current decision because of the change of the pheromone. The process of ACO includes four basic steps: initializing trail, local trail update, analyzing tours, and global trail update. In [D04], the authors analyzed the convergence properties of the ACO, and proved that under certain conditions, the ACO can always guarantee convergence.

PSO is a method which simulates the social behaviour of birds to search the solution for an optimization problem. In the PSO, each candidate solution in the search space is called a particle and each particle has a vector which determines the flight direction and distance. The quality of the particle is estimated by the fitness value. The particles find the solutions by following the particle that has the highest fitness value. A typical PSO consists of four parts: initialization, evaluation, comparison, and change of particle status. Since the particles update their status according to the current best solution, the PSO always converges very quickly towards the optimization position.

In the PEA, each processor initializes a subpopulation which consists of individuals based on the global information. The EA is executed on each processor to find the global solution. After communication between different processors, the individual with the highest value is the final result. According to the communication topology of the processors, PEA can be classified into three classes: 1) Master-Slave [CG00], 2) Multiple-Deme [HZD07], 3) Fine-Gained [KKM00]. In Master-Slave PEA, a

population is initialized at a master processor, and the individuals in the population are assigned to slave processors. Then evaluation of the individuals and the application of EA operations are repeated in parallel at the slave processors. Final result is selected by the master processor after it collects the individuals with the highest value of fitness from the slave processors. In multiple-deme PEA, each processor initializes a subpopulation that contains a set of individuals to execute the evolution operations respectively and a subset of individuals will be selected with a random probability. Then the selected subset will be exchanged among the neighbours to compare the fitness value of the individuals. The above steps are repeated until the final result is obtained. In fine-gained PEA, each processor initializes an individual, and then the evolution operations are executed by communicating with a set of neighbouring processors.

Table 3.1: The differences between PEA and LEA

PEA	LEA
Individual is encoded by using the global information.	Individual is encoded by using the partial information stored on the processor.
The size of each individual on different processors is the same.	The size of each individual on different processors can be different.
The solution obtained on each processor is the global result for the whole problem.	The solution obtained on each processor is the result for the sub-problem.
The individual with the highest fitness value is selected as the final solution.	Processors communicate with each other to assemble/elect the best individual as the final solution.

Although the above methods can speed up the convergence or reduce the computational time, all of them are based on the assumption that the global information has been collected and stored at each processor. In this paper, the

proposed LEA is inspired by PEA and also borrows the idea of PEA in part of the framework. However, LEA is different from PEA in four main aspects as shown in Table 3.1. For distinguishing PEA, traditional EAs and LEA, Figure 3.1 shows the differences of the definitions of the population and subpopulation among these algorithms. We can see that the population in traditional EAs is divided into subpopulations in PEA and LEA. In the PEA, the subpopulation is a crosscut subset of the population, while it in the LEA is a straight cut subset of the population.

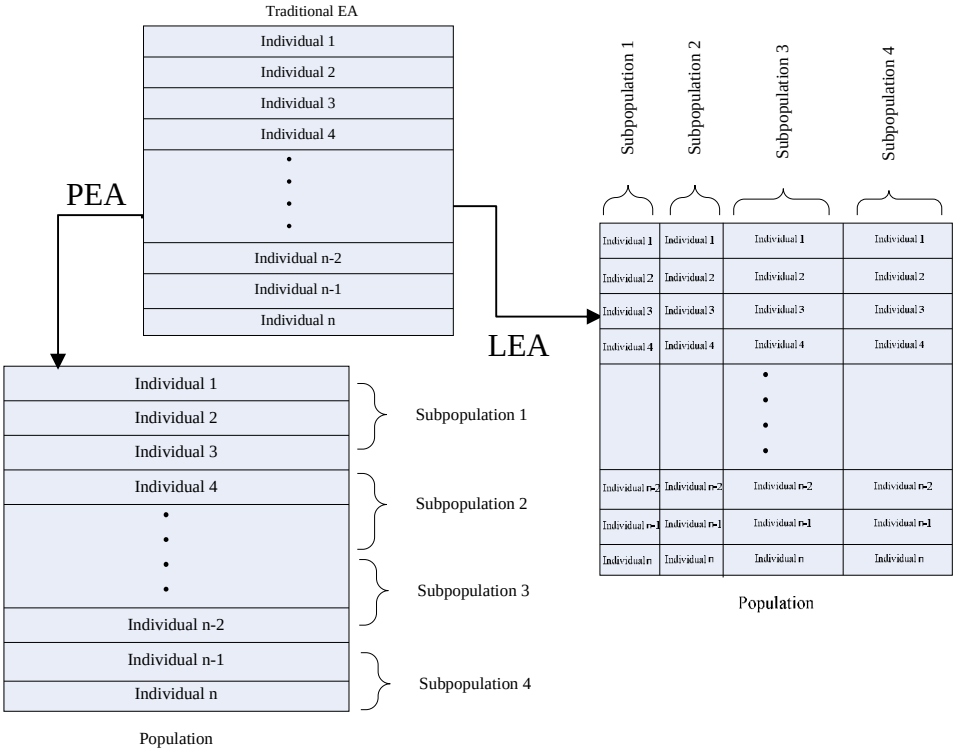

Figure 3.1: The differences between PEA and LEA

3.2. The Localized Evolutionary Algorithm Framework

Before we present LEA, the definitions are clarified which may be different from the traditional EAs.

Definition 1. (Local information): Local information is defined as that information is distributed on a processor based on its own knowledge.

Definition 2. (Global information): Global information is the collection of local information from all the available processors.

Definition 3. (Local solution): Local solution is the solution of a subproblem which has been obtained on a processor.

Definition 4. (Global solution): Global solution is a solution for the whole problem.

Definition 5. (Search space): Search space represents all the possible global candidate solution .

Definition 6. (Sub-search space): Sub-search space represents all the possible local candidate solution for a subproblem.

Definition 7. (Subpopulation): Subpopulation means the set of possible local candidate solution on a processor.

Definition 8. (Individual): Individual is a local candidate solution.

Definition 9. (Overlapping search space): Overlapping search space is a common subset of variables between the sub-search spaces.

The LEA consists of four parts: initialization, local search, coordination, and global evaluation. The subpopulation on each processor is randomly initialized in the beginning. Then, EA operations are executed to determine the optimal local solution on each processor through local search. During executing the above operations, cooperation between processors is implemented through coordination. Finally, global evaluation is carried out to deal with local solutions and obtain the global optimal solution. Figure 3.2 shows that there are 5 processors which can communicate with

each other and the LEA is executed on each processor. The search space consists of 5 sub-search spaces. Each processor holds the local information and finds a local solution in a sub-search space. Therefore, each processor creates a subpopulation based on its information. The dashed line means that there is overlapping search space between a pair of processors and coordination is executed between the processors. The solid line means that the two processors can communicate with each other and global evaluation is executed between the processors. Figure 3.3 depicts the procedure of the LEA on each processor, which includes initialization, local search, coordination, and global evaluation. The dash line between coordination and initialization or local search part means coordination can be inserted into both of them.

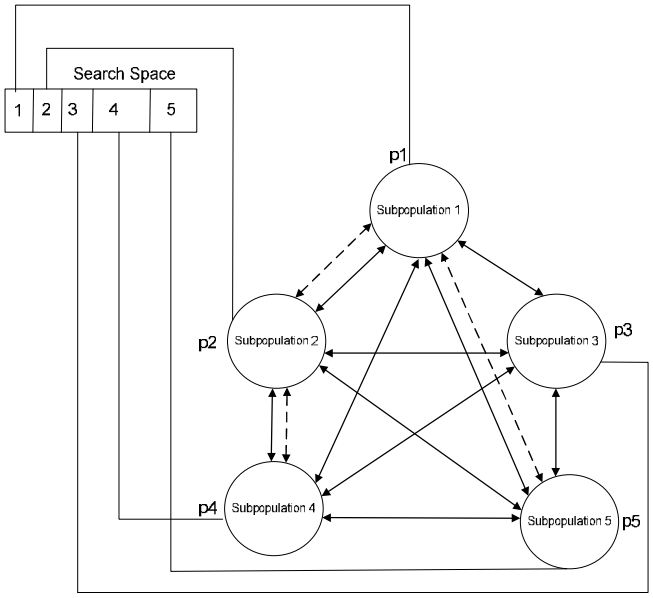

Figure 3.2: The example of the LEA executed on five processors

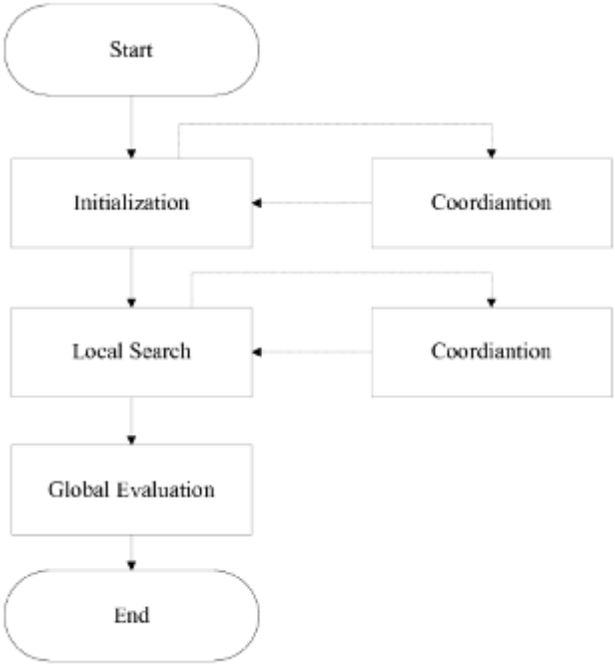

Figure 3.3: The strategy for the LEA. Initialization, local search and coordination are executed one time which is called one round

3.2.1. Initialization

In the initialization stage, the subpopulations are created $P = \{pop_1, pop_2, ..., pop_m\}$on each processor. The number $|pop_i|$ of individuals in processor i can be different, $i = 1, 2, ...,m$. m is the number of subproblems. The variables in each individual can be dynamically changed according to the feedback from the other processors (see part 3.2.3: Coordination).

3.2.2. Local Search

A local search is implemented on each processor to solve the subproblem. The basic procedures for each processor are similar, and their major differences are in the design of the fitness functions and the coding of individuals.

Table 3.2: Notation

Symbol	Descriptions
POP_i	The subpopulation
P	The populations
P_i	The processor i
T_w	Time interval
N_p	The size of population
N_i	The number of individual of subpopulation
N_v	The number of variables of individual
Gen	The maximum number of generations
P_c	The probability for crossover operation
P_x	The probability for mutation operation
C_i	The individual i
OS_{wp}	The optimal solution of the whole problem
OS_i	The optimal solution of the i^{th} subproblem
N	The number of processors
M	The size of the used memory
G	The counter of generations
Ser	The size of the search space
S_i	The set of processor in the cluster i
R_i	The request sent by P_i
M_i	The message sent by P_i
R_c	The communication range
R_s	The sensing range
$avgG$	The average number of generations

Step 1: The fitness of each individual in the subpopulation is evaluated in each processor.

Step 2: For each processor, recombination and mutation are performed.

Step 3: The fitness of the offspring is evaluated. The evaluation depends on the result of other processor (see part Coordination).

Step 4: The best-ranked individuals are selected to reproduce in each node and the worst-ranked individuals in the subpopulation are replaced by the offspring.

These steps are repeated until a termination condition is satisfied, e.g., time limit or sufficient fitness has been attained.

3.2.3. Coordination

There are two kinds of operations in the coordination. The first is that processors exchange partial local information to obtain their optimal local solutions. The second is that the idle processors help the busy processors to obtain the local solution. Since each processor holds only limited local information, the number of variables in the sub-search space may be more than those in the local information. Thus, in order to obtain the optimal solution for subproblem, it is essential to propose an operation to exchange the partial local information of the processors. The following two guidelines are used for coordination:

1) Is it necessary to increase or reduce the variables on a processor? This is related to the impact of the variables on the optimal result in a processor.

2) Is it necessary to know the fitness values of other processors so as to evaluate the fitness of another processor? This is related to the impact of the fitness values of another processor on the fitness of another processor.

Based on the above considerations, we propose two operations.

Operation 1 - If variables from other processors can be assigned to another processor and lead to significant improvement in the quality of the solution for that processor, the variables of the individuals in the corresponding processors can be dynamically adjusted.

The adjustment is considered in the initialization part. There are two types of processors for the adjustment: the master processor and the ordinary processor. All the variables in the master processor are in the overlapping search space which can be assigned to the corresponding ordinary processors. In the ordinary processor, the individual in the subpopulation is initialized by the local information and variables from the corresponding master processor. The duty of the master processor is to communicate with the related ordinary processors, assign its variables to the ordinary processors, and execute a local search to find the optimal assignment of the variables. The duty of the ordinary processor is to carry out local search to obtain the local solution. The essence of the adjustment operation is to adjust the local information to obtain a better solution.

Operation 2 - If the fitness value of the processor P_A is affected by other processors, the processor PA calculates the fitness of each individual in the subpopulation after collecting the fitness values of individuals from other processors.

Suppose that the fitness value $f_{p_A}(C_i)$ of processor P_A is affected by processors P_B and P_C. The fitness value $f_{p_A}(C_i)$ is calculated as follows:

$$f_{p_A}(C_i) = \alpha f_{p_B}(C_j) + \beta f_{p_C}(C_h) + \lambda \quad \textbf{(3.1)}$$

Similarly, $f_{p_B}(C_j)$ is the fitness value of processor P_B, $f_{p_C}(C_h)$ is the fitness value of processor P_C. Parameters α and β are priority factors for processors P_B and P_C, respectively. C_i, C_j and C_h are individuals in P_A, P_B, and P_C respectively. λ is a constant value. In the above case, when the fitness of each individual of PA needs to be evaluated, P_A needs to communicate with P_B and P_C to obtain the fitness values of the corresponding individuals in P_B and P_C.

It is critical to note that the above operations are only two examples in the first kind of coordination. These operations can be inserted into initialization and local search. More coordination operations can be carried out according to the specific applications. Since the processor calculates the result according to its own subpopulation respectively, the computational time needed varies. If the size of individual in the subpopulation is small, the processor quickly finishes the calculation. It will take a longer time if the size of the individual in the subpopulation is larger. To speed up the computation of the whole system, the basic idea of PEA is borrowed, that is, calculation is conducted in parallel way. There are two types of processors to execute the second kind of coordination: helper and requester. The detailed procedures are as follows.

1) When the processor obtains the local solution, it becomes idle. The idle processor sends an idle message to the neighbors.

2) When the process is calculating the solution, it is busy. If a busy processor receives an idle message from its neighbor, it checks its status. If the current index of generation G, $Gen- G < G_t$ or the current result closes to the desired result, the processor does nothing. Otherwise, it becomes a requester and sends a help message

to idle processor. G_t is a threshold to determine whether the processor requests or does not request for help.

3) The idle processor receives the help message, then becomes the helper and replies to the first coming help message.

4) The requester waits for T_w time interval to receive the replies and divides the current subpopulation into *ns* subsets. Each subset consists of N_p/n_s individuals. The number of subsets is decided upon by the number of helpers received in a time interval T_w. The subsets of the subpopulation are sent to the helpers respectively.

5) When these helpers receive the partial subpopulation, they begin to search for the solution in parallel and the status of helpers become busy.

6) When these helpers finish searching, the individual with the highest fitness value is sent to the corresponding requester. Then, the status of the helper becomes idle again.

7) The requester receives the results from the helpers and compares the candidate solutions to select the optimal solution.

The steps are repeated when help messages are sent by the busy processors. We can say that these cooperating processors are working in parallel. More idle processors mean that more processors can search for the solution for another processor at the same time. Figure 3.4 is an example of the first kind of coordination. There are four processors and they can communicate with each other. In (1), the four processors are calculating their own local solutions. Their status is busy. (2) When Processor 1 becomes idle, it sends idle messages to other three neighbors. (3) Processor 2 and 4 send help messages to Processor 1. (4) Processor 1 sends the reply to Processor 2. (5) There are 8 individuals in the subpopulation on Processor 2. Processor 2 divides

the subpopulation into 2 subsets that each subset includes 4 individuals and sends a subset of subpopulation to Processor 1. The status of Processor 1 and Processor 2 become busy. (6) Processor 1 and 2 search for the solution for Processor 2 at the same time. (7) When Processor 1 finds the solution, it sends the result message which involves the individual with highest fitness value to Processor 2. (8) Processor 2 compares the solutions and finds the optimal solution. (9) Processor 1 and Processor 2 send the idle messages to their neighbors respectively. Figure 3.5 shows the concrete procedures of initialization, local search and coordination.

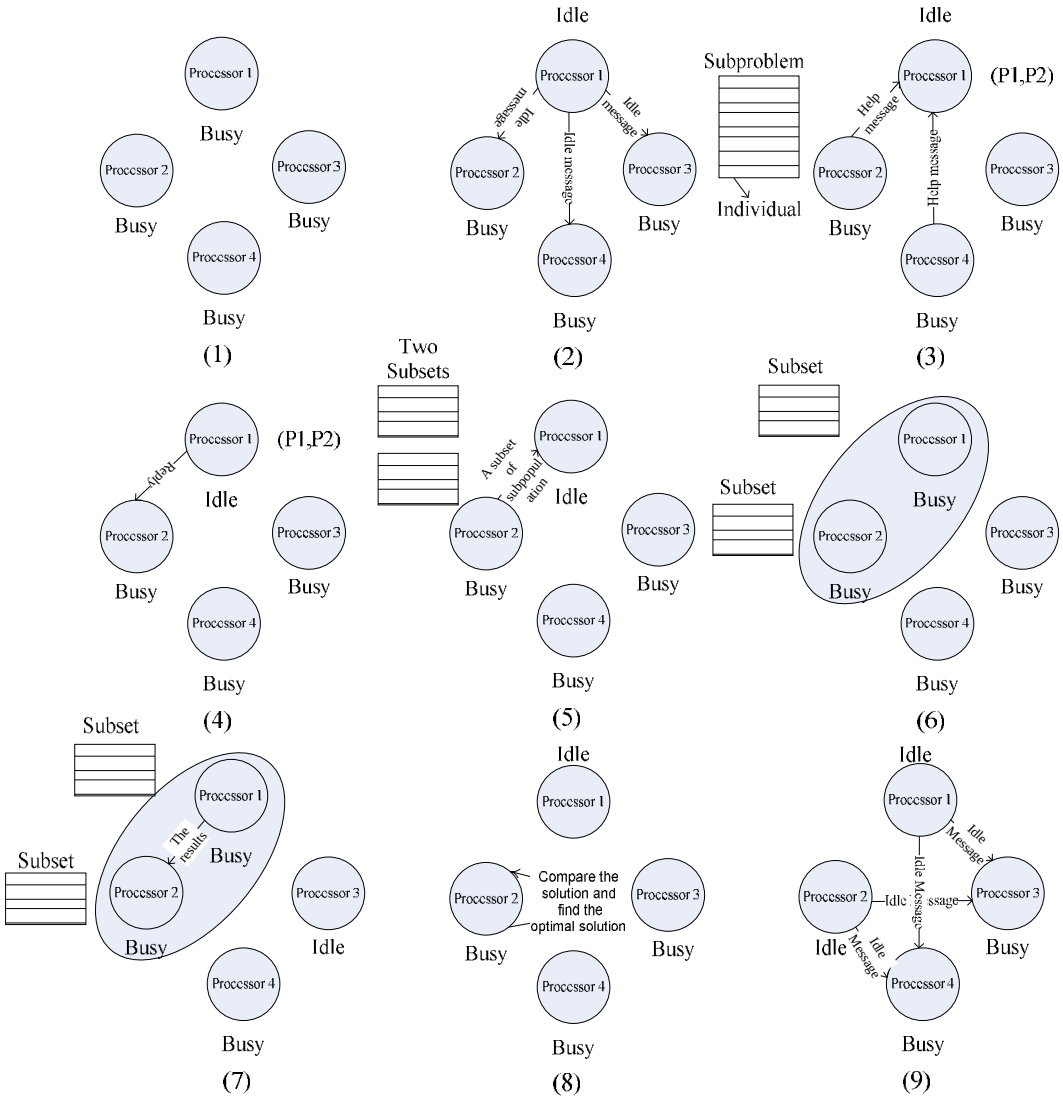

Figure 3.4: An example process of the second aspect in coordination part

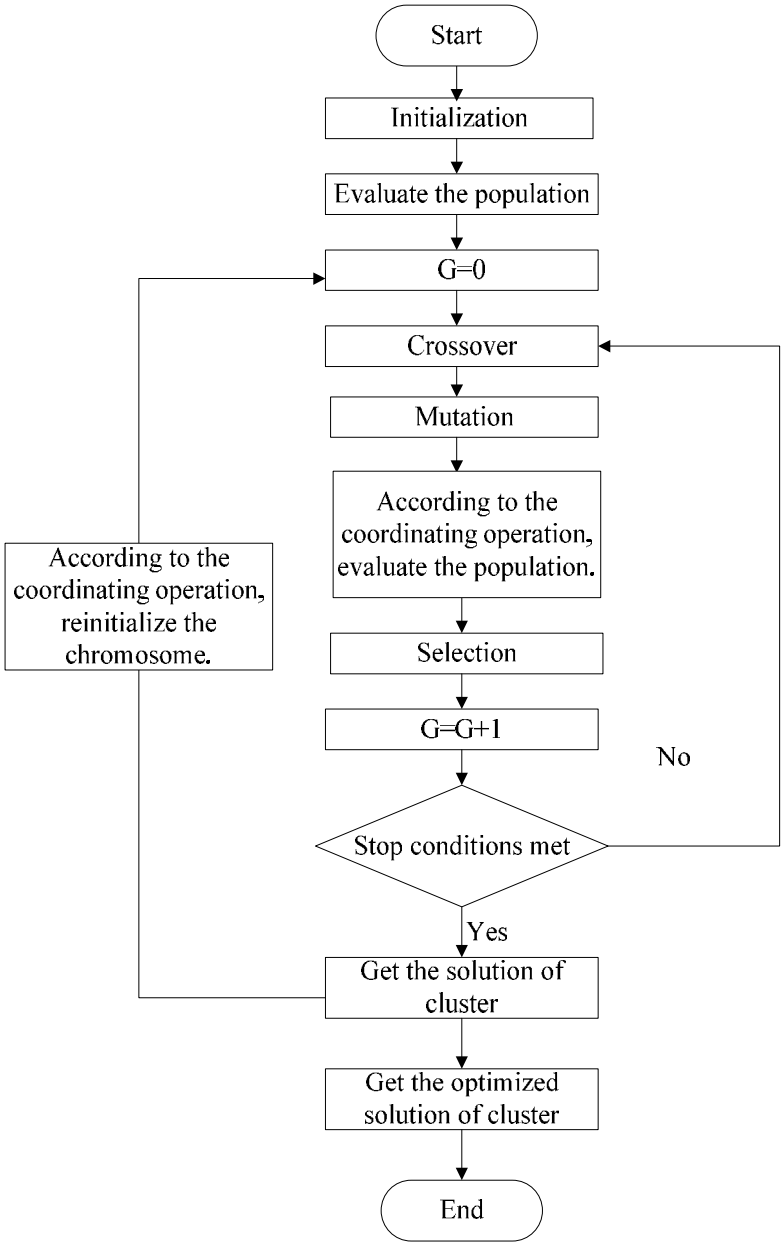

Figure 3.5: The procedures of initialization, local search and coordination, where G represents the generation index

3.2.4. Global Evaluation

Global evaluation is performed after the local solutions of the processors are obtained. In order to obtain the global solution of the whole problem, the processors

need to communicate with neighboring processors. At the same time, operations such as comparison or combination between the local solutions of the processors are implemented during the communication. The framework of the LEA is given in **Algorithm 1** as shown in Figure 3.6. A simple example that explains the process of obtaining the global solution from the local solutions is presented as follows.

Suppose that the problem is divided into m subproblems, the solution for the problem is defined as:

$$OS_{wp} = combination(OS_1, OS_2, \dots, OS_m) \quad \textbf{(3.2)}$$

where OS_{wp} is the global optimal solution of the problem, and OS_i is the solution of i^{th} subproblem. For attaining OS_{wp}, the detailed evaluation steps are as below:

Each processor P_i maintains an item I. Initially, I is set to false.

1) If P_i finishes the calculation of a solution, it sends a request R_i to all of its neighboring processors.

2) When P_j receives a message R_i from P_i, it sends a reply to P_i, showing its status of item I.

3) When P_i receives a reply from P_j , it checks the value of item I of P_j .

4) If $I = true$, it does nothing. Otherwise, it sends the message M_i, including its solution, to P_j.

5) When P_i has sent M_i to all the neighbors, its item I is set to true.

6) When P_j receives a message M_i from P_i, it integrates its solution OS_j with P_i's solution OS_i and packages the integration into M_j.

7) The above steps are repeated until all the node items are set to true. The global result is thus obtained.

Algorithm 1: The Localized Approach for Evolutionary Algorithms

while the subproblem *i* needs to be solved **do**
  $G = 0$
  Initialize $POP(G)$
  $f(POP(G)) = f(C_1(G)), f(C_2(G)), \ldots, f(C_n(G))$
  Evaluate $POP(G)$
  while not termination condition 1 **do**
    $POP_C(G)$ = crossover $POP(G)$
    $POP_m(G)$ = mutation $POP_C(G)$
    Evaluate $POP_m(G)$ with coordination operation
    $POP(G + 1)$ = select[$POP_m(G)$];
    $G = G + 1$
    if termination condition 1 & not termination condition2 **then**
      $G = 0$
      Initialize $POP(G)$with coordination operation
    end if
  end while
  Solution = C_{best}
  Communicate with other processors
  $OS = C_{best} \cup OS$
end while
Evaluate *OS*

Figure 3.6: The localized approach for EA

3.3. Evaluation of the Framework

3.3.1. Finding the Optimal Number of Subproblems

The number of subproblems is determined by the property of the problem, the scale of the search space, the number of processors, and the knowledge of the processors. If the problem is divided into too many subproblems, the communication time between the processors will increase. If the number of subproblems is small, the computational time will increase. Therefore, how to obtain the number of subproblems that can minimize the implementation time is a challenge. Suppose the implementation time of the LEA is the sum of the computation and communication time:

$$T_P = n_d \times g \times T_f + (a + b) \times T_C \qquad \textbf{(3.3)}$$

where g is the number of generations which is dependent on domain, T_f is the time of executing the fitness estimation, and T_c is the time to communicate with one of the neighboring nodes, n_d is the size of population, a is the time that coordination are executed between processors. b is the time that global evaluation is executed between processors.

Assume that the scale of the whole search space S_{er} is fixed, the number of subproblems is h. And also assume that the number n of variables in the individual on each processor is the same which is equal to N/h and the value of each variable is equal to 0 or 1. Therefore, the number of possible solutions (individuals) is equal to 2^n, then n_d is:

$$n_d = \gamma \times 2^{\mathrm{n}} \qquad \textbf{(3.4)}$$

where$\gamma$is the probability of the individuals involved in the subpopulation. Since the coordination operations and global evaluation depend on the number of subproblem$h$, $a = \emptyset \times h, b = \delta \times h$. $\emptyset$ is the probability that a processor needs to cooperate with other processors during coordination. δ is the probability that a processor needs to communicate with other processors during the global evaluation. Therefore, function (3.5) can be changed to (3.6):

$$T_P = \gamma \times 2^{\frac{N}{h}} \times g \times T_f + (\varphi + \delta) \times h \times T_C \quad \textbf{(3.5)}$$

then, we obtain the first derivation and the second derivation of (3.5).

$$\frac{\partial T_P}{\partial h} = ln2\gamma \times 2^{\frac{N}{h}} \times g \times T_f \times N \times \left(-\frac{1}{h^2}\right) + (\varphi + \delta) \times h \times T_C \quad \textbf{(3.6)}$$

$$\frac{\partial^2 T_P}{(\partial h)^2} = ln2\gamma \times 2^{\frac{N}{h}} \times g \times T_f \times N \times \left(N \times ln2\frac{1}{h^4} + \frac{2}{h^3}\right) \quad \textbf{(3.7)}$$

Because $\frac{\partial^2 T_P}{(\partial h)^2}$, $\frac{\partial T_P}{\partial h}$ is monotone increasing, when h=*2*, if $\frac{\partial T_P}{\partial h} > 0$, T_p is monotone increasing which is shown in Figure 3.7. The optimal number of subproblems is 2. Thus when the number of subproblems is 2, the execution time is minimum. When h=*2*, if $\frac{\partial^2 T_P}{(\partial h)^2} < 0$, the value of T_p is shown in Figure 3.8. From the figure, we find that there exists a peak value h_0 which is the optimal number of subproblems. Thus when the number of subproblems is h_0, the execution time is minimum.

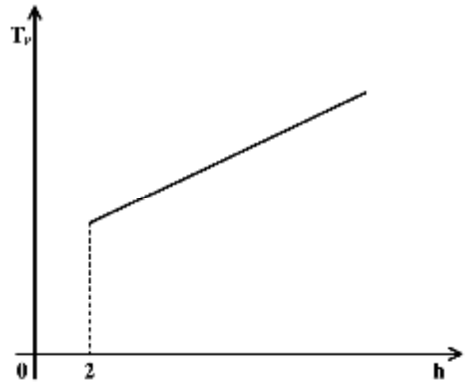

Figure 3.7: The curve of T_p when $h=2$, $\frac{\partial T_P}{\partial h} > 0$

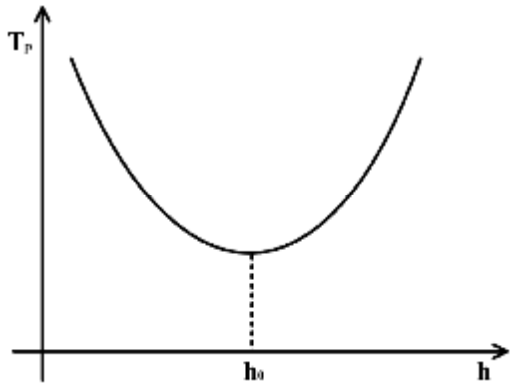

Figure 3.8: The curve of T_p when $h=2$, $\frac{\partial^2 T_P}{(\partial h)^2} < 0$

3.3.2. Evaluation Metrics of the LEA

In this section, we use three metrics to estimate the performance of the LEA. The first metric is time T_{mt} which consists of the time needed for collecting the information and computing the final result.

$$T_{mt} = t_{cl} + t_{cm} \qquad \textbf{(3.8)}$$

The process of collecting the information includes releasing the requirements and collecting the information. The time of collecting the information t_{cl} is mainly the communication time. t_{cl} is increased as the increase of the size of system and the distribution of the information. The time of computing the final result t_{cm} includes the time of executing the LEA and the time of communication.

The second metric is the quality of the final result Q_{re}. According to the objective of an optimization problem, the quality of the result is measured by the fitness value. The third metric is the consumed storage of the processor S_{tore}, which is estimated by the size of the subpopulation stored in a processor. Assume there are N_i individuals in the population, and there are N_v variables in the individual, the size of each variable is B_s, then S_{tore} can be calculated as

$$S_{\text{tore}} = N_i \times N_v \times B_S \quad \textbf{(3.9)}$$

3.4. Application for the Framework

3.4.1. The Suitable Applications

The LEA can be applied to many systems to solve optimization problems. However, the proposed algorithm is not suitable for all scenarios and all problems. It is for solving an optimization problem that 1) can be divided into subproblems, 2) the subproblems should have the same optimization objectives, 3) there is no exact sequence between the subproblems. In this section, two types of appropriate problems that can be solved by LEA are discussed.

3.4.1.1. Combinatorial Problem

A combination is an arrangement in which the order does not matter [I08]. The objective of the problem is to find the optimal combination to satisfy various

constraints. The problem can be separated into several subcombinatorial problems. When using the LEA to solve the problem, coordination operations should be inserted into initialization in order to adjust the variables of an individual.

3.4.1.2. Graph Partitioning Problem

Given a graph, the graph partitioning problem [HK00] means dividing a graph into pieces, such that the pieces are about the same size and there are few connections between the pieces. Since the graph can consist of subgraphs, the problem also can be divided into subproblems. The objective of the subproblem is to divide the subgraph into pieces. When using the LEA to solve the problem, exchanging local information with the corresponding processors before a new round of LEA to obtain the optimal solution is the main operation of coordination. For example, in a monitoring system, each camera captures an image of a limited area. When processing the whole image, at first, the image needs to be divided into segments. When each processor attached on the camera uses the LEA to execute image segmentation, the processor needs to exchange partial information with the related processors during coordination. And the whole image segmentation is attained by combining the segmentations of processor in the global evaluation.

3.4.2. Energy-Efficient Coverage Problem in WSNs: An Example

3.4.2.1. Description of the Algorithm

In this section, an energy efficient coverage problem is used as an application example for studying the effectiveness of the proposed LEA framework. Suppose that there are K sensors $S = \{s_1, s_2, ..., s_k\}$ deployed in a given area A randomly. Each sensor has an identical sensing range R_S and an identical communication range R_C ($R_C > 2R_S$). Each sensor knows its locations L_{Si}. The objective of the problem is to

maximize the number of the disjoint cover sets which can completely cover the area *A*, and to identify the corresponding sets of sensors. Suppose area *A* is initially divided into two subareas, see Figure 3.9(b), and the sensors in the subareas and on the split lines are grouped respectively. The original problem is divided into 3 subproblems that are to maximize the number of disjoint cover sets which can completely cover the subareas. For example, in Figure 3.9(c), the sensors in the two subareas and those near the midline form three groups, U_1, U_2, and U_3, with processors h_1, h_2, and h_3 respectively.

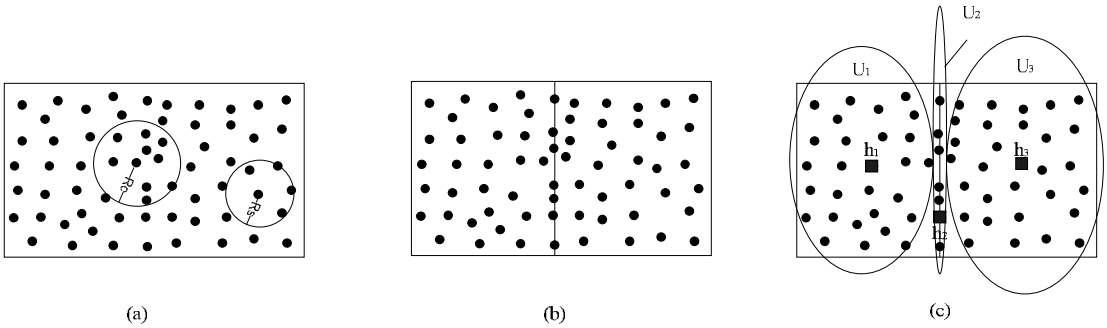

Figure 3.9: Illustration of the sensors in the area A. (a) The network (b) The network is divided into two subareas (c) The network after clustering

According to the proposed LEA framework, the duty of h_2 is to assign its group members to other two groups, U_1 and U_3, and determine the optimal assignments of the sensors in the group to maximize the number of cover sets, and the duty of h_1 and h_3 is to obtain the maximum number of cover sets and the related sets in the subarea.

The coordination part is inserted into the initialization. h_2 first initializes the subpopulation in the group U_2. The number of variables in the individual is equal to the number of the group members. The value of each variable is equal to the ID of the subarea, the ID of U_1 is 1, the ID of U_3 is 2. Suppose that there are 7 sensors in U_2, Figure 3.10 illustrates an individual in the subpopulation of U_2. Then, the sensors

of U_2 are assigned to U_1 and U_3 according to the values of variables. Figure 3.11 shows the subarea after reassignment of the variables. h_1 and h_3 also initialize the individuals in which the variables are the sensors in their subarea and the sensors assigned by h_2.

2	1	2	2	2	1	2

Figure 3.10: An individual in cluster U_2

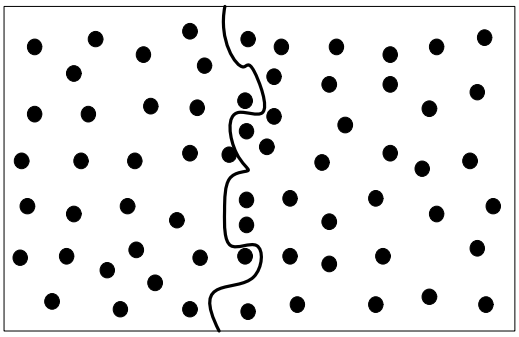

Figure 3.11: The subarea after reassignment the variables according to the individual in U_3

In the local search, h_1 and h_3 calculate the number of disjoint cover sets and the corresponding sets by using the proposed EA algorithm. The coordination in local search is to evaluate the fitness value of individuals cooperatively. h_1 and h_3 send messages which include the number of disjoint cover sets and time stamp to h_2. h_2 receives the messages and selects the minimum number of disjoint cover sets with the same time stamp as the fitness of individual.

In the global evaluation, individuals with the highest fitness value in U_2 are the final optimization result. Figure 3.12 shows the LEA steps among the three processors. ①

The population is initialized in h_2. ②The variables are assigned to groups U_1 and U_3 according to the individual in U_2. ③The EA operations are executed on U_1 and U_3. ④ The solutions are sent to h_2. ⑤The solutions are compared.

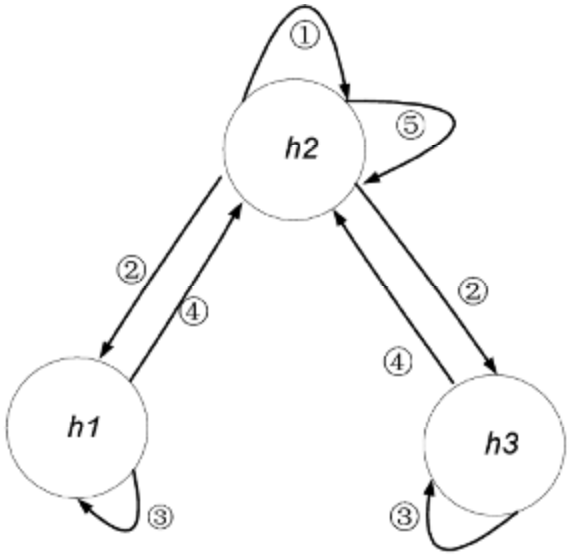

Figure 3.12: The steps among the three processors

3.4.2.2. Simulation Results

The proposed LEA is tested with different sensor deployments. There are two factors which affect the deployment: the number of sensor nodes in the network and the sensing range of the sensors. The performance of the LEA regarding different numbers of subareas is compared with existing EA algorithms, multiple-deme PEA, genetic algorithm for maximum disjoint set covers (GAMDSC) [LTK07], markov chain monte carlos (MCMCCs) [SP01]. For the sake of comparison, the sensing ranges or the number of nodes is fixed. The algorithm will terminate when the result stops changing or the counter of generation is equal to the maximum number of generations G=*Gen*. In the simulation, p_c= 0.1, p_x=0.04. LEA2, LEA4, and LEA8 mean the LEA is executed based on 2, 4, 8 subareas respectively. PEA2 and PEA4

mean the PEA is executed based on 2, 4 processors. At first, LEA is compared with multiple-deme PEA which is very similar to LEA.

We assume that each processor initializes N_i individuals in subpopulation, and each individual holds N_v variables. If the size of a variable is 1 bit, the size of the used memory is $S_{tore} = N_i \times N_v$. From Figure 3.13, in PEA, we find that the size of the storage consumed will not change even the number of processors increases. However, in LEA, Store will decreases as the number of subareas increases. And obviously, the used memory in LEA is much smaller than that in PEA.

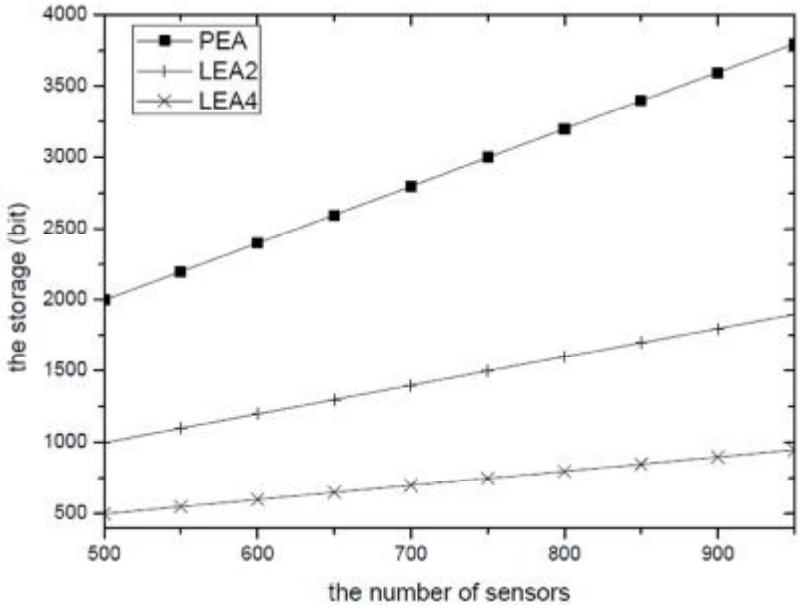

Figure 3.13: Comparisons of the size of consumed storage in each processor

For further comparison, the numbers of disjoint sets obtained in PEA and LEA are compared. We fix the sensing range $R_s = 15$, and vary the number of sensor from 500 to 950. Figure 3.14 shows that, in PEA with 2 and 4 processors, the numbers of complete cover sets obtained are the same. However, in LEA2, the optimization result is equal to the result of PEA in 7 cases. As to computation time, when using

the same number of processors or subareas to execute PEA and LEA, the computation time of LEA is much shorter than PEA in Figure 3.15.

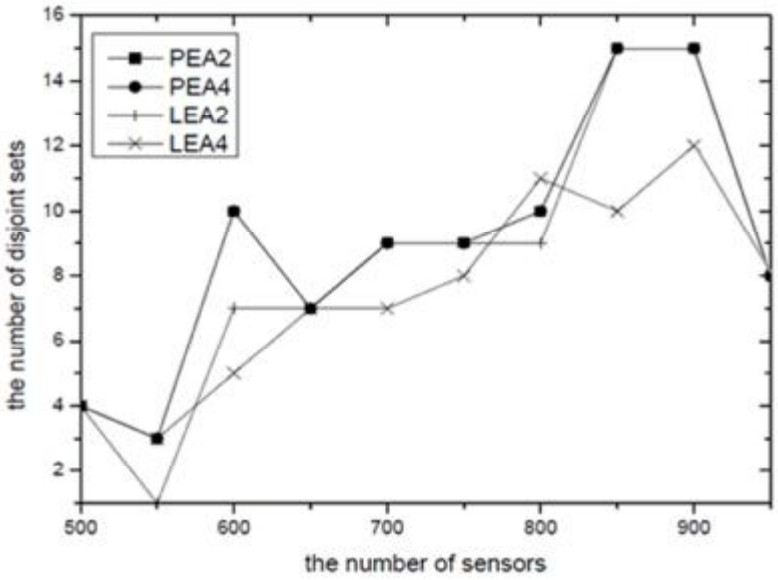

Figure 3.14: Comparisons of the number of sensor nodes against the number of disjoint cover sets when sensing range R_s= 15

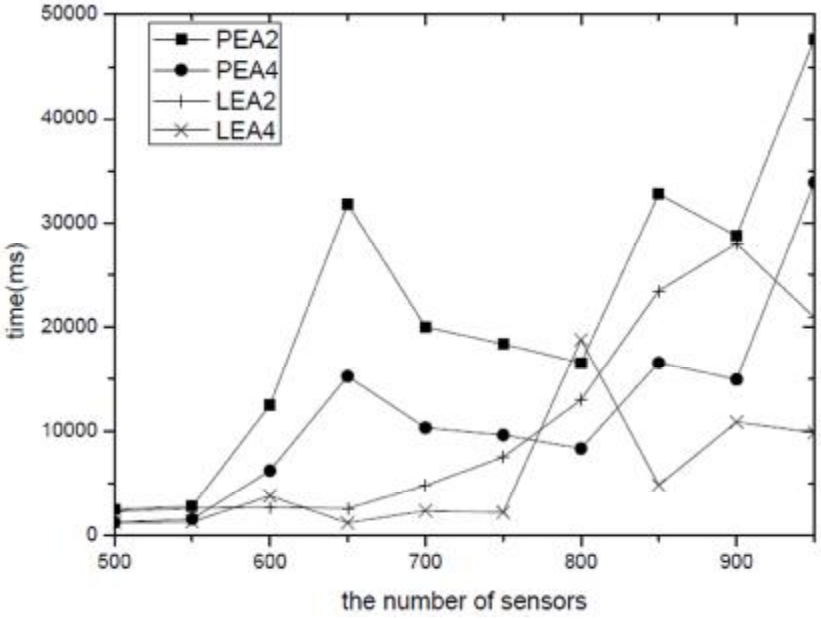

Figure 3.15: Comparisons of time against the number of sensors using when sensing range R_s= 15

Then, we will discuss the influence of the number of subareas on the computation time and quality of the LEA. Figure 3.16 compares the number of disjoint cover sets obtained by the proposed LEA which is based on 2, 4, 8 subareas, GAMDSC, and MCMCCs. The sensing range is set from 6 to 24 for the test case N = 300. From the figure, we find that when the area is divided into two subareas, the proposed LEA achieves a result that is equal to the result of the GAMDSC in 5 cases. The result of the proposed LEA is better than the result of the GAMDSC in 2 cases. Moreover the result of the proposed LEA can achieve 4 out of the 5 results in MCMCC. On the other hand, when the number of subareas increases, the quality of the result is reduced. The time used by the LEA is much shorter than that of the GAMDSC and MCMCCs in all 10 cases which is shown in Figure 3.17. When the number of subareas increases, the used time decreases.

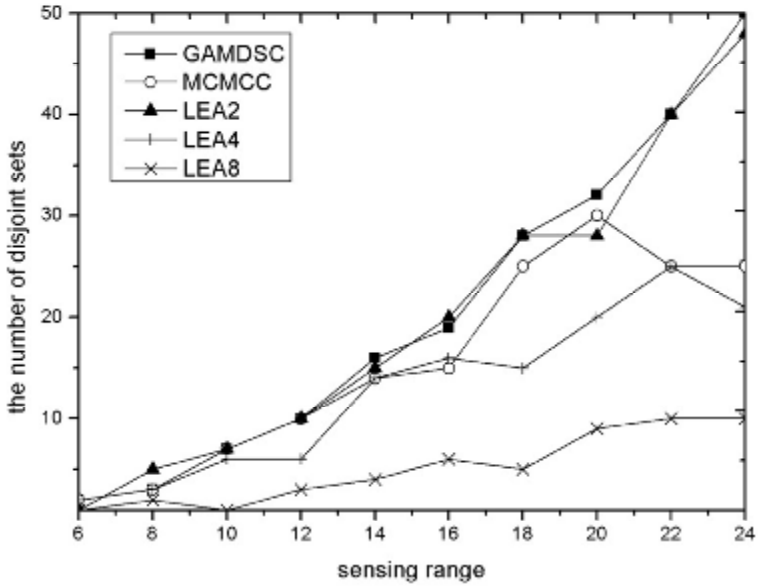

Figure 3.16: Comparisons of the number of disjoint cover sets against the sensing range when number of sensors N=300

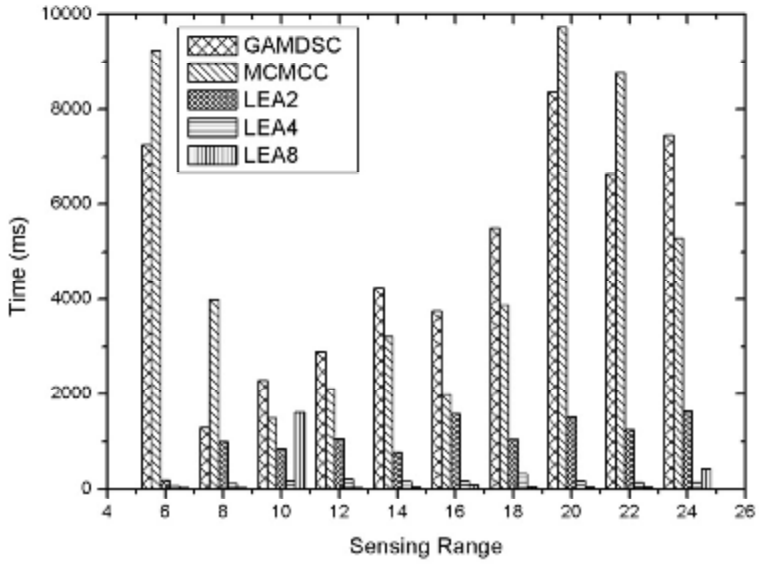

Figure 3.17: Comparisons of the time against the sensing range when number of sensors N=300

Figure 3.18 shows the number of disjoint cover sets when there are N sensors in the network and N increases from 500 to 950, R_s=8. The final results of the LEA based on 2, 4, 8 subareas, and GAMDSC, MCMCCs are the average number of disjoint complete cover sets that are obtained by computing 100 independent runs. From N=650 to N=950, the values obtained by the proposed LEA based on 2 subareas is equal to that of the GAMDSC. From N=600 to N=950, the performance of LEA2 is better than that of the MCMCCs. Figure 3.18 also shows the quality of the result for LEA4 and LEA8 is worse than LEA2. Figure 3.19 shows that the LEA saves more time than the GAMDSC and MCMCCs.

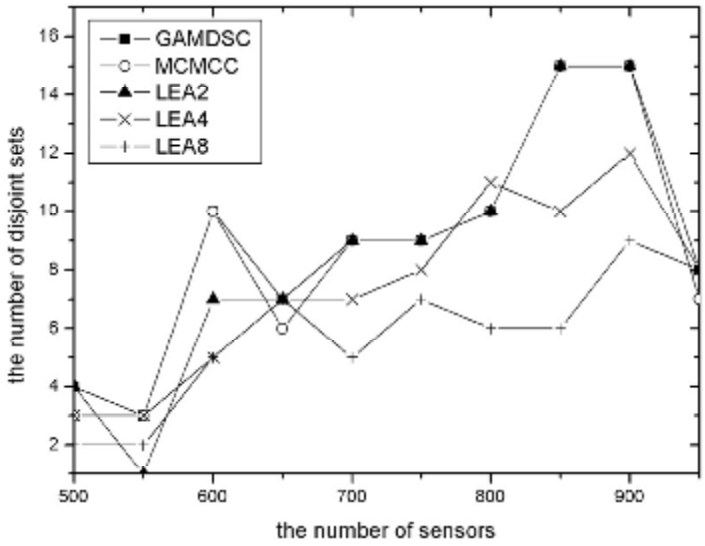

Figure 3.18: Comparisons of the number of disjoint cover sets against the number of sensors using fixed sensing range R_s=8

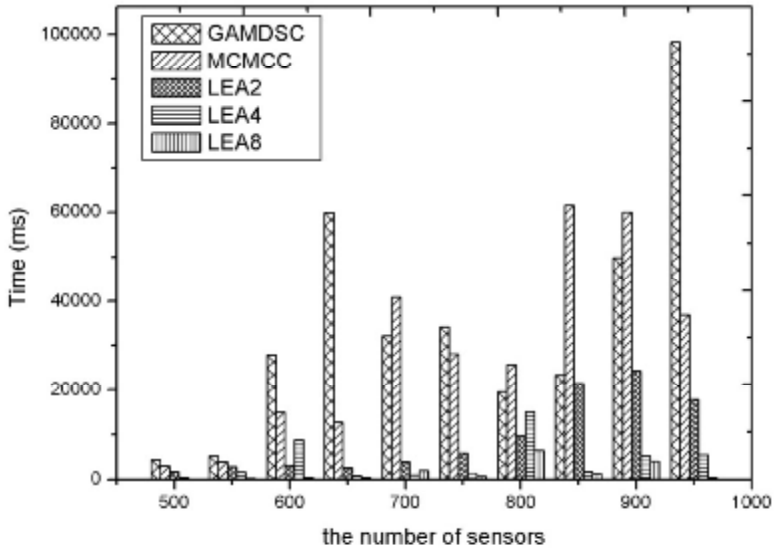

Figure 3.19: Comparisons of the time against the number of sensors using fixed sensing range R_s=8

3.5. Summary

In this chapter, a localized approach for evolutionary algorithms (LEA) has been proposed. The framework consists of four parts, i.e., initialization, local search, coordination, and global evaluation. In the initialization, the subpopulation is initialized at each processor and the parameters of the EA are also initialized. Then, a local search and coordination are executed to search for the local solutions. The optimization solution of the whole problem is obtained after the global evaluation. LEA is used for solving the problem that can be divided into subproblems with same optimal objectives and without exact sequence among them. An example of using the LEA to maximize the number of complete cover sets to prolong the lifespan of WSNs is studied. The results are favourably compared with centralized EA approaches and PEA, the consumed time and the storage of LEA are much shorter/smaller than these algorithms. In the future work, we will investigate more applications of LEA.

Chapter 4. Coverage-Preserving Routing Algorithms for Wireless Sensor Networks

In this chapter, we present the proposed coverage-preserving routing algorithm for WSNs. At first, section 4.1 overviews the work. The preliminary and the definition of coverage preserving routing problem (CPRP) are given in section 4.2. In section 4.3, we first propose how to calculate the sensing areas of multiple sensors. And then section 4.4 presents the detailed distributed and centralized solutions for the CPRP. Section 4.5 describes the proposed algorithms that can be applied to the irregular sensing regions. The simulation and discussion of the proposed algorithm are presented in section 4.6. Finally, section 4.7 concludes the chapter.

4.1. Overview

Recent years have witnessed a growing interest in the application of WSNs. Examples of such applications include environmental monitoring, health, object tracking, intelligent transport system, etc. In these applications, effectively monitoring the events of interest and reliably delivery the collected information to sink in a timely manner are two basic tasks. These two tasks are associated with sensing coverage and routing of WSNs, respectively. Considering the network scalability and limited energy of wireless sensor nodes, both of them need to be implemented in an energy-efficient way.

Although both sensing coverage [MKP05] and routing [AK04] have been studied extensively, there still lacks of researches combining these two issues together. Considering the fact that a node in a WSN can take the responsibility of both sensing and data delivery, combining sensing coverage together with routing will have some extra benefits. For example, if the overall sensing area of the nodes in a routing is maximized, fewer number of remaining nodes in the area need to be functioned to

cover the area. The energy consumption of whole network can therefore be saved. Another advantage of a larger sensing area of the nodes in the routing is that the probability that a new event will be detected by these relay nodes is increased. The new event detected by these relay nodes can be directly sent to the sink along this path without the necessity of establishing a new one. For example, in monitoring of fire in a forest, if the newly developed fire point is covered by the sensor nodes in a routing path, the newly sensed data can be sent along the existing path quickly without finding a new path. Also, in applications of object tracking in WSNs, if the trajectory of the object can be covered by the relay nodes of routing, the sensed data will be sent to sink in real time without finding a new routing.

A simple example is illustrated in Figure 4.1. Assume the maximum number of hops from the source to sink is required to be four. Under this constraint, two paths are selected and are illustrated in Figure 4.1(a) and Figure 4.1(b), respectively. Although both of these two paths are of the same length, the sensing areas of the sensor nodes in these two paths are different. The path selected in Figure 4.1(a) is better that shown in Figure 4.1(b) since more areas are covered by the nodes in this path.

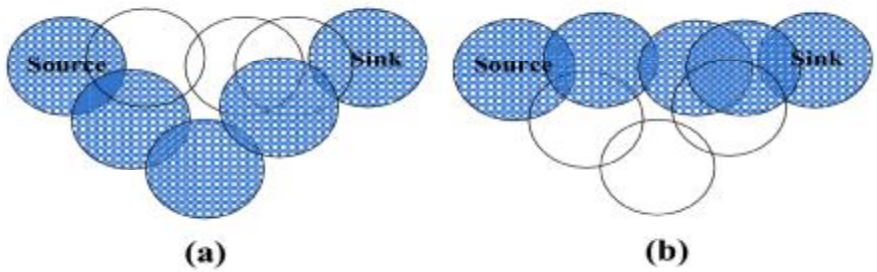

Figure 4.1: Examples of choosing routing with(a)/without(b) considering sensing coverage

In the chapter we first define the coverage-preserving routing problem (CPRP) in the WSNs. Then we design a method based on Monte-Carlo integration to approximately calculate the sensing area. Based on the method, two algorithms, one distributed and the other centralized are proposed. Compared with the existing works, we have the following contributions:

- We propose an efficient while light-weighted method to calculate the total sensing area of multiple nodes, particularly when the sensing areas overlap with each other.

- We propose two algorithms to maximize the sensing coverage of routing under delay constraint.

- The proposed algorithms do not need the location information. The performance of our algorithm is demonstrated through simulation data.

4.2. Preliminaries and Problem Definition

4.2.1. Sensing Coverage of Routing

In this section, we will define the overlapping area as well as the sensing coverage of a routing.

Definition 1. (Sensing coverage of routing): Let the routing $R = (v_0, ..., v_n)$ consists of n nodes, the sensing disk of the sensor v_i is S_{v_i}. The sensing coverage of routing is

$$S_R = \bigcup_{j=1}^{n} S_{v_i} \qquad \textbf{(4.1)}$$

Figure 4.2 illustrates the sensing coverage of the routing $R=(v_0, v_1, v_2, v_3)$ $S_R = S_{v_0} \cup S_{v_1} \cup S_{v_2} \cup S_{v_3}$. Since the sensing areas of randomly deployed sensors may partially

overlap, if the sensing range of sensor v_i is $R_s^{v_i}$, the overlapping area O_R of N_R nodes in the routing R can be calculated as:

$$O_R = \sum_{j=1}^{N_R}(\pi(R_s^{v_i})^2) - S_R \qquad \textbf{(4.2)}$$

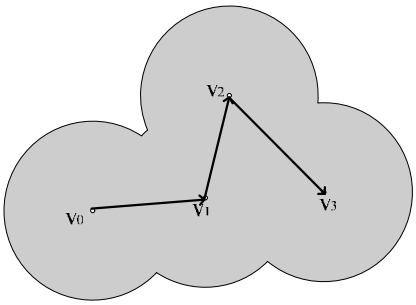

Figure 4.2: The sensing coverage of routing

4.2.2. Problem Definition

It is assumed that there are N sensors randomly deployed in a given area A. Each sensor has sensing range $R_s^{v_i}$and $R_s^{v_i}$ can be different. The communication range of the node R_c is identical which is $R_c \geq 2maxR_S^{v_i}$. The location of the sensors are unknown, but the distance $d(v_i,v_j)$ of each neighbor v_i, v_j can be estimated by radio. The directional information (the orientation of each neighbor) $\theta_{v_iv_j}$ is also a prior knowledge which can be obtained by antenna. $\theta_{v_iv_j}$ is the angle between node v_i and node v_j. There is a sink node and a source node, the source node sends the sensing data to the sink. The objective of the problem is to find the routing with maximum sensing coverage and the constraint that the hop of the routing should be less than K. We call this problem as Coverage-Preserving Routing Problem (CPRP). The problem formulation is:

Let G be an undirected graph $G=(V,E)$, where $V=\{v_0,...,v_N\}$ is the set of nodes, $|V|=N$, E is the set of edges, s and t are two specified nodes. If $d(i,j) < R_c$, there is an edge between the node v_i and v_j. The sensing range of each node is S_{v_i}. The objective of the CPRP is to find a simple path $R=(v_0,...,v_n)$ from s to t in G, where $v_0=s$, $v_n=t$, $(v_{g-1}, v_g) \in E$ for all g from 1 to n, such that

$$max \bigcup_{i=0}^{n} S_{v_i} \qquad \textbf{(4.3)}$$

subject to $n-1 \leq K$

4.2.3. The Hardness of CPRP

In this section, we give below a brief formal proof of the complexity of CPRP.

Theorem 1: The Coverage-Preserving Routing Problem in wireless sensor networks is NP-hard.

Proof: The problem can be reduced to a weight constrained shortest path problem (WCSPP) which is a well know NP-Hard problem [DB03]. For proving the hardness of the CPRP, *effective coverage* is defined.

Definition 2. (Effective coverage): Let the routing $R=(v_0,...,v_{i-1},v_i)$, $v_0=s$. Assume a set Γ_{EC} of nodes v_{i-j} is the set of neighbors of node v_i in the routing R, where $j \leq i$, $|\Gamma_{EC}| = N_{EC}$, $N_{EC} \geq 1$. The effective coverage of node v_i corresponded to R is

$$EC_{v_i}^{R} = S_{v_i} - \bigcup_{h=1}^{N_{EC}} (S_{v_i} \cap S_{v_\mu}) \qquad v_\mu \epsilon \Gamma_{EC} \qquad \textbf{(4.5)}$$

The sensing coverage of routing can be regarded as cumulation of effective coverage of the nodes in the path. The Eq.4.1 can be transferred as

$$S_R = S_{v_0} + \sum_{i=1}^{n} EC_{v_i}^{R} \qquad \textbf{(4.6)}$$

If we use the effective coverage of the node as the weight of the edge between the node and the previous node in the corresponding routing, the problem formulation of the CPRP can be transferred as follows.

Let G be an undirected graph $G' = (V', E')$, where $V' = \{v_0, \ldots, v_n\}$ is the set of nodes, $|V'| = N$, E' is the set of edges, s and t are two specified nodes. Each edge $(v_i, v_j) \in E$ has a weight $w_{v_i v_j}$ which is effective coverage of node v_j, and an associate time cost $c_{v_i v_j}$. The value of $w_{v_i v_j}$ related to different path is different, the value of $c_{v_i v_j}$ is equal to 1. The objective of the CPRP is to find a simple path R' from s to t in G', where v_0=s, v_N=t,$(v_{g-1}, v_g) \in E'$ for all g from 1 to n, such that

$$max \sum_{e \in E''} w_e \qquad \textbf{(4.7)}$$

$$\text{subject to } \sum_{e \in E''} c_e \leq K \qquad \textbf{(4.8)}$$

$$c_e = 1, \forall e \in E'' \qquad \textbf{(4.9)}$$

where E'' denotes the set of edges in the path R', i.e.,$E'' = \{(v_{g-1}, v_g) | v_g \in R'\}$.

According to the above problem formulation, we find that it is similar to the WCSPP but it is harder than WCSPP because of the complex computation for the effective coverage. Thus, the CPRP is also a NP-hard problem.

4.3. Calculating the Sensing Coverage of Routing

Since there are many types of overlapping area of different nodes, it is impossible to compute the sensing area of routing accurately by only using the directional and distance information. Thus, we use a proposed method to approximately calculate the areas. The method is derived from Monte-Carlo integration in [KW08] which means using the random points to estimate the area. The basic idea of computing the

sensing area is to uniformly deploy the N_p points in sensing area of each node. Then the sensing area is calculated by checking the point to see if it is within the other circles. The calculation is regarded as a preprocessing when the nodes are deployed, the output of which is a matrix for each node.

Considering real environment, we assume that there is no location information available for all nodes because of the expensiveness of GPS. Thus computing the sensing coverage of routing exactly is impossible. Moreover, because of the limited computation ability for sensors, the computation executed in each sensor should be as simple as possible. Hence, in the proposed algorithm, the distance and orientation of each neighbour and the modified Monte-Carlo integration are used as a substitute to approximately estimate the sensing coverage of routing.

Assume there are $N_p^{v_i}$ points uniformly distributed in the sensing area of a node v_i. The density of the points deployed in the node is predefined as D_p, thus $N_p^{v_i} = S_{v_i} / D_p$. All the IDs of the neighboring nodes are recorded in each node. Regarding the location of node as (0,0), for each node, the basic steps of the preprocessing are:selecting a neighbouring node v_j as the basic benchmark, generating the axis system that the positive direction of the x-axis is the direction of the line segment from v_i to a neighbouring node v_j, the vertical direction of x-axis is the y-axis, the N_p coordinates of points are generated according to the coordinate system shown in Figure 4.3, transforming coordinates from node v_i to any neighbouring nodes, obtaining the overlapping matrix and storing the matrix in the node. The details of the coordinate transformation and the matrix generation are described as follows.

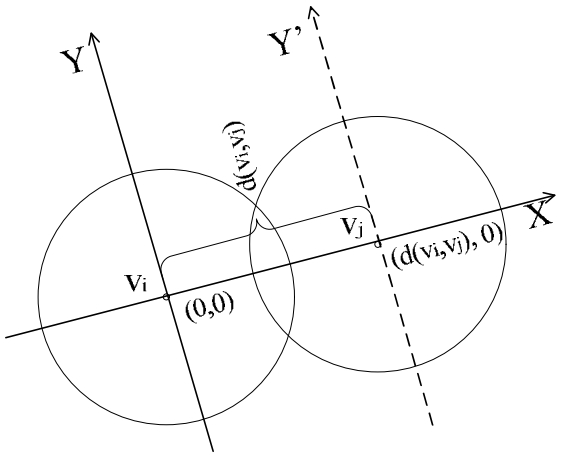

Figure 4.3: Transforming the coordinate of the points from v_i to v_j

4.3.1. Coordinate Transformation

There are two parts of the transformation. One of the parts is horizontal moving of the coordinates. The other part is rotation of the axis system.

In first part, we only need to use Eq.4.10 to transform the coordinate of points in the axis system of node v_i to the coordinate in the axis system of node v_j.

$$\begin{cases} X'_{P_i} = X_{P_i} - d(v_i, v_j) \\ Y'_{P_i} = Y_{P_i} \end{cases} \quad \textbf{(4.10)}$$

where X_{P_i} is the coordinate of point p_i on x-axis, andY_{P_i} is the coordinate of point p_i on y-axis. X'_{P_i} and $Y'_{P_i}$are the transformed locations.

In second part, as shown in Figure 4.4, for node v_h, we cannot simply transform the coordinate of the points of node v_i by Eq.4.10. For obtaining the location of points according to the coordinate system that the direction of x-axis follows line segment from node v_i and v_h, we need to rotate the current axis. The axis rotatory angle is the

angle between line segment $L_{v_i v_j}$ and $L_{v_i v_h}$, which can be denoted by φ. Because the orientation of each neighbour is known, φ is computed as:

$$\varphi = \theta_{v_i v_h} - \theta_{v_i v_j} \qquad \textbf{(4.11)}$$

Therefore, the coordinate of point p_i is transformed as:

$$\begin{cases} X''_{P_i} = X_{P_i} cos\,\varphi + Y_{P_i} sin\,\varphi \\ Y''_{P_i} = -X_{P_i} sin\,\varphi + Y_{P_i} cos\,\varphi \end{cases} \qquad \textbf{(4.12)}$$

After axis revolving, according to Eq.4.10, the coordinate of points in the axis of node v_h can be obtained.

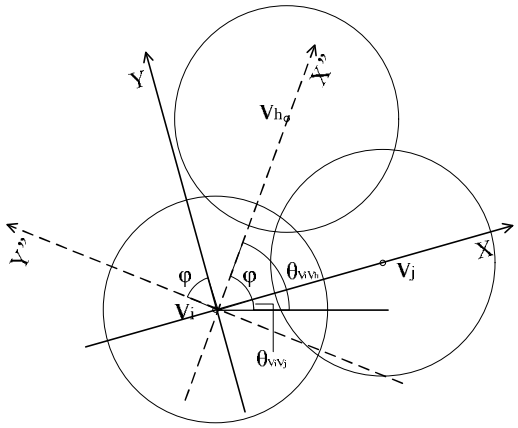

Figure 4.4: Rotating the coordinate system with φ

4.3.2. Obtaining the Matrix

After transforming the location of points, we need to check if the point is within the sensing range of neighboring node. We use the Euclidean distance to obtain the distance between the point and the related neighboring node.

$$Len = \sqrt{X'^2_{P_i} + Y'^2_{P_i}} \qquad \textbf{(4.13)}$$

$Len \leq R_s$ means point p_i is in overlapping area between two nodes. We use a matrix M to record the points shown in matrix (4.14).

$$M_{v_i} = \begin{pmatrix} X & Y & v_1 & v_2 & & v_m \\ X_1^{v_i} & Y_1^{v_i} & 0/1 & 0/1 & \cdots & 0/1 \\ X_2^{v_i} & Y_2^{v_i} & 0/1 & 0/1 & & 0/1 \\ \vdots & & & & \ddots & \vdots \\ X_{N_{P-1}}^{v_i} & Y_{N_{P-1}}^{v_i} & 0/1 & 0/1 & & 0/1 \\ X_{N_P}^{v_i} & Y_{N_P}^{v_i} & 0/1 & 0/1 & \cdots & 0/1 \end{pmatrix} \qquad \textbf{(4.14)}$$

The first and second column records the coordinates of points. The elements from third column to the last column record if the corresponding point is within the overlapping area between node v_i and all the neighboring nodes. The value of elements can be equal to 0 or 1 means the point is both in node v_i and the corresponding node. 1 means the point is not in the corresponding node.

The above operations repeat until all neighbors of the node have been considered. Finally, the matrix records the relationship of multiple sensing areas. The sum α of the elements in column corresponded to node v_j means that the effective coverage of node v_i, which corresponds to the node v_j, is α. The number β of elements in row corresponded to the point p_i means that the point p_i is not covered by β corresponding

nodes. If β is equal to the number of neighboring nodes of node v_i, the related point is only covered by node v_i. The process of obtaining the matrix in each node can be regard as the preprocessing when the nodes are deployed in the area.

4.4. Coverage-Preserving Routing Algorithms

In this section, we will describe a centralized and a distributed coverage routing algorithm to solve CPRP.

4.4.1. A Distributed Algorithm Based on LS for Coverage-Preserving Routing Problem

In this section, we present a distributed algorithm to solve CPRP. Based on the matrix, the proposed algorithm is modified from LS algorithm. In LS algorithm, each sensor has a label to record the cost from sink to the current node [AMO93]. The labels will be changed during the iterative processing until the desired path has been obtained. It is a greedy algorithm which can obtain the routing through the local dynamic change of labels.

Since WCSPP can be solved by LS algorithm, the LS algorithm can also be used for CPRP. However, in CPRP, the value of $w_{v_iv_j}$ between the node v_i and v_j in the different routing will be different which is the key difference from WCSPP. For example, in Figure 4.5, there are two different routing R_1=(v_0,v_1,v_2,v_3), R_2=(v_0,v_2,v_3). The weight of the edge (v_2,v_3) is $w_{v_2v_3} = S_{v_2v_3} = S_{v_3} - S_{v_2} \cap S_{v_3}$ in R_2. However, in R_1, the weight of the same edge (v_2,v_3) is $w_{v_2v_3} = S_{v_2v_3} = S_{v_3} - (S_{v_2} \cap S_{v_3}) \cup (S_{v_1} \cap S_{v_3})$. Based on the above reason, we need to modify the LS algorithm. The detailed algorithm is presented in this section.

Let each node have a set of labels in which each label relates to a different path from the sink t to that node. And each label has two numbers denoted as $(W_{v_i}^g, C_{v_i}^g)$, where

$W_{v_i}^g$ and $C_{v_i}^g$ represents the sensing coverage and the length of g routing respectively. Each label must record the label with different value of sensing coverage of routing. The $W_{v_i}^g$ and $C_{v_i}^g$ of the node v_i can be computed as $W_{v_i}^g = W^g{}_{pre_{v_i}} + w_{pre_{v_i}v_i}$, $C_{v_i}^g = C_{v_i}^g + 1$,where pre_{v_i} is the previous node of node v_i in the g path.

Definition 3. Let $(W_{v_i}^g, C_{v_i}^g)$ and $(W_{v_i}^q, C_{v_i}^q)$ be two labels related to two different path g q at node v_i. We say that $(W_{v_i}^g, C_{v_i}^g)$ is smaller than $(W_{v_i}^q, C_{v_i}^q)$if and only if $W_{v_i}^g \geq W_{v_i}^q$, $C_{v_i}^g \leq C_{v_i}^q$.

Definition 4. A label is said to be *possible* if there is no other labels is smaller than the label at the node.

Definition 5. A routing g is said to be *feasible* if the corresponding label is possible at each node of the routing and $C_s^g \leq K$, where C_s^g is the time cost of $(W_{v_{i-1}}^g, C_{v_{i-1}}^g)$ on the source.

Definition 6.A routing g is said to be *best* if it is feasible and W_s^g is smaller than the weight of any label on the source.

The proposed algorithm is to find the possible label on each node and the best routing from sink to source. It consists of two operations: 1) Computing the weight of the edge; 2) Updating the labels.

4.4.1.1. Computing the Weight of the Edge

In each node's array, the sum of a column (except the first and second column) represents the approximate size of the effective coverage of the node for the neighbors corresponded to the column. When the node v_i receives the query from the node v_j, it checks the column the node v_j corresponds to in its matrix. It also checks its neighbors whether the neighboring node is already selected in the routing. If its

neighboring node v_h is the previous node of the routing, the overlapping area of the routing will increase. Thus, node v_i should check the column which the node v_h corresponds to in its matrix. It should be noted that the elements is counted only if the value of elements corresponded to neighboring nodes of the routing in a row is equal to 1. The counting result is the weight of the edge (v_i, v_j).

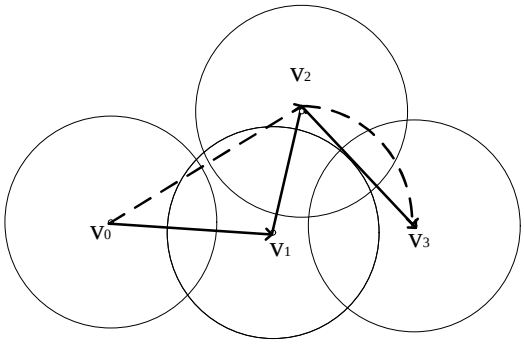

Figure 4.5: Two different routings

For example, in Figure 4.5, if the node $v_1v_2v_3$ as a routing $R=(v_1,v_2,v_3)$, the weight $w_{v_1v_2}$, $w_{v_2v_3}$ can be calculated by using the matrix. To calculate $w_{v_1v_2}$, node v_2 checks its matrix M_{v_2} shown in matrix 4.15. In M_{v_2}, the third column which is related to node v_1 is checked. The sum of the column's elements is 4, thus $w_{v_1v_2} = 4$. For $w_{v_2v_3}$, node v_3 checks its matrix M_{v_3} shown in matrix 4.16. Since the previous node v_1 of the path and the sending node v_2 are the neighbors of node v_3, the columns corresponded to the nodes are checked. According to the above rules, we can see that the elements of column v_1 and column v_2 in row 0, row 1, and row 5 are 1. Thus, $w_{v_2v_3} = 3$.

$$M_{v_2} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cccccc} X & Y & v_1 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \begin{pmatrix} 1.21 & 2.11 & 1 & 1 & 0 & 1 & 1 \\ 3.12 & 1.98 & 1 & 1 & 0 & 1 & 0 \\ 4.23 & 9.73 & 0 & 1 & 1 & 0 & 1 \\ 12.4 & 1.00 & 1 & 0 & 1 & 1 & 1 \\ 7.92 & 10.64 & 1 & 1 & 0 & 0 & 0 \\ 4.56 & 3.54 & 0 & 0 & 0 & 0 & 0 \\ 9.45 & 7.08 & 0 & 1 & 1 & 0 & 1 \\ 10.00 & 4.00 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \quad \textbf{(4.15)}$$

$$M_{v_3} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cccccc} X & Y & v_1 & v_2 & v_7 & v_8 \end{array} \\ \begin{pmatrix} 5.66 & 6.44 & 1 & 1 & 0 & 1 \\ 4.77 & 0.89 & 1 & 1 & 0 & 1 \\ 9.33 & 9.73 & 0 & 1 & 1 & 0 \\ 1.35 & 1.00 & 1 & 0 & 1 & 1 \\ 10.00 & 6.00 & 1 & 0 & 0 & 0 \\ 1.24 & 3.01 & 1 & 1 & 0 & 0 \\ 9.45 & 4.00 & 0 & 1 & 1 & 0 \\ 8.23 & 9.20 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \quad \textbf{(4.16)}$$

4.4.1.2. Updating the Labels

At first, there is no labels on any node except for the label (*S*,0) on sink *t*. *S* is the sensing range of sink, $S=N_p$. Sink sends the query to the neighbors at the start. The node v_i receiving the query adds the label $(W_{v_i}^g, C_{v_i}^g)$ to $Label_{v_i}$, where $W_{v_{i-1}}^g = S + w_{v_i v_j}$, $C_{v_{i-1}}^g = 1$, $g \in I_{v_i}$, I_{v_i} is the index set of labels, and$Label_{v_i}$is a set of labels on node v_i. Then the neighbors broadcast the message with the label and the query. The node v_j receiving the message processes the received label $(W_{v_i}^g, C_{v_i}^g)$, $W_{v_j}^g = W_{v_i}^g + w_{v_i v_j}$, $C_{v_j}^g = C_{v_i}^g + 1$. And the received label after processing is compared with the stored labels in $Label_{v_j}$. If one of the labels in $Label_{v_j}$is smaller than the received label or $C_{v_j}^k > K$, it should be dropped. Otherwise, it is a possible label that will be stored in $Label_{v_j}$. The operation will repeat until the source receives all the possible labels. Finally, the best routing is obtained at the source node. The proposed algorithm, as it applies to the CPRP, is given in **Algorithm 2** as shown in Figure 4.6.

Algorithm 2: The distributed coverage-preserving routing algorithm

INPUT:

a matrix ***M***, the number ***Neg*** of neighbors, the number ***Nre*** of received message, the received label l_{v_i} from node v_i, a set ***Label*** of labels, a set ***R*** of routing, $i \in I$, ***I*** is the index of labels

BEGIN

Preprocessing

if $C_{v_i}^g + 1 \leq K$ **then** Calculating $w_{v_i v_j}$

for (*j*=0; *j*<|*I*|; *j*++) **do**

if l_{v_i}in $Label$ that $l_{v_i} > (W_{id}^g, C_{id}^g)$ **then**

$$Label = Label \cup (W_{id}^g, C_{id}^g)$$

Update ***I***

endif

endfor

endif

Nre=***Nre***+1; Broadcast the (W_{id}^g, C_{id}^g) and the query

if ***Nre***=***Neg***+1 and $W_{v_k}^g$ is the minimal in $Label$ **then**

r_g in ***R*** is the best routing

endif

End

Figure 4.6: The distributed coverage-preserving routing algorithm

4.4.2. A Centralized Algorithm Based on GA for Coverage-Preserving Routing Problem

Genetic algorithms (GAs) are search techniques which simulate the process of biological evolution. In this paper, we use GA to find the path from source to sink with maximum sensing coverage and the hop of the path is less than *K*. The first hurdle of using GA is working out how to best encode the possible solutions as genes. In the current problem, we use the IDs of the nodes to encode the possible solutions as genes. The first position and last position of the encoding are the IDs of the source node and the sink respectively. The middle positions are the IDs of the nodes that are selected as the nodes of the routing path. For example, in the encoding '458270', the ID of source node is '4', and the ID of the sink node is '0'. '5', '8', '2', '7' are the IDs of selected nodes in the path. Since the number of nodes in the routing path will be different as the different paths are selected, the length of encoding cannot be predicted. According to the constraints of hops, the path length of CPRP must be equal or smaller than *K*. Therefore, the length of path is predefined as *L* and *L=K*. According to the representation of the possible solution, the detailed steps of GA are as follows.

4.4.2.1. Initialization

At first, *L*2 nodes are randomly selected from the network. Since the randomness of the initialization, there exists invalid encoding which means the nodes of the chromosome are not connected from source node to sink. The example of invalid chromosome is shown in Figure 4.7(b), the dotted line means there is not the connected edge in the network between the nodes that are in the chromosome. For obtaining the valid tour (show in Figure 4.7(a)), a CHECK scheme is proposed to delete the invalid ones.

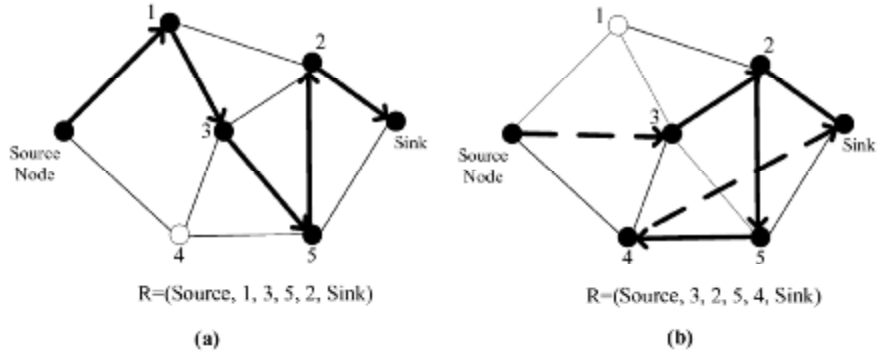

Figure 4.7: An example of initialization of an individual

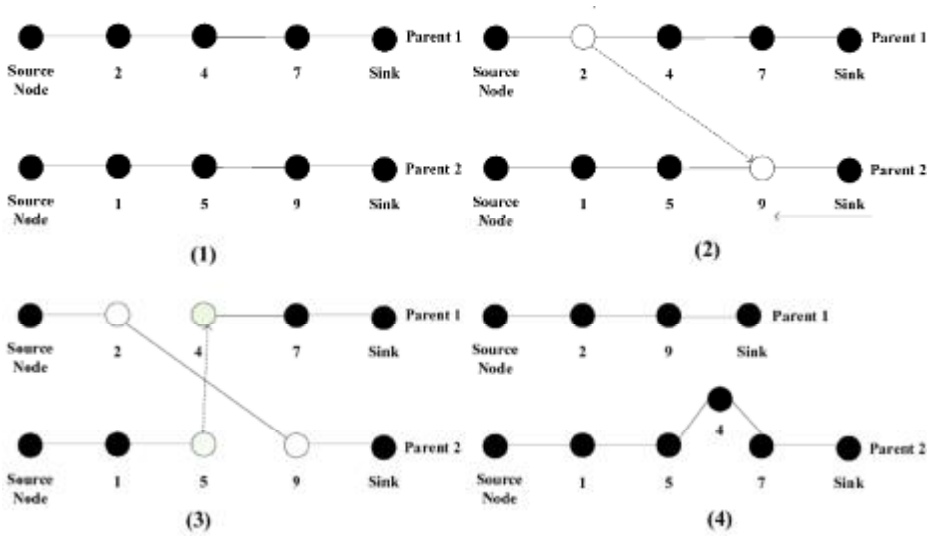

Figure 4.8: An example of the crossover operation

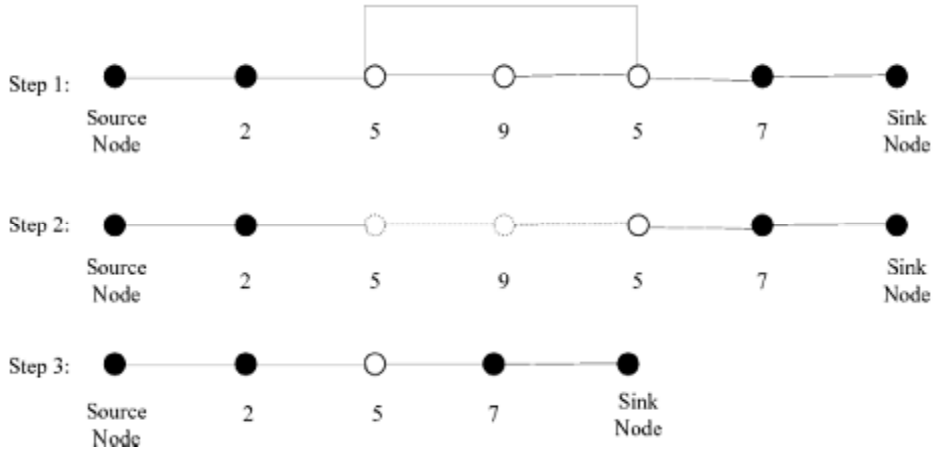

Figure 4.9: The operations of deleting the repeated genes

4.4.2.2. Crossover

If the traditional crossover technique is adopted, many chromosomes will be generated. Thus, another repair scheme is used to guarantee the validity of offspring. The process is described in Figure 4.8. Step 1, there are two parents randomly selected from the population. Step 2, a random position in the parent is selected (except the first and last position). For example, if the position is '3', as to the parent 1, the second position from source to sink is the crossover point. As to the parent 2, the second position from sink to source is the crossover point. Then if the crossover point of the parent 1 can connect with the crossover point of the parent 2, the new offspring is created. In the step 3, if the forward position next to the crossover point in the parent 1 can connect with the backward position next to the crossover point of the parent 2, the other new offspring is also created in the step 4. However, the individual after crossover suffers the problem as follows. 1) There are repeated IDs of nodes in an individual. 2) The length of individual is greater than the predefined length. For the first problem, the middle nodes between two repeated nodes and a

repeated node are deleted. Figure 4.9 shows the process of the method. As to the second problem, the invalid individuals are deleted.

4.4.2.3. Mutation

We design a technique to ensure that the chromosome is valid after the mutation. The detailed steps are 1) selecting an individual randomly from the population; 2) generating a random mutation point in the individual; 3) for the selected node that is needed to mutate, selecting a neighboring node. If the neighboring node is connected with previous and next node of mutation point in the individual, it will replace the mutation point. And the new offspring of the individual is created. Figure 4.10 illustrates an example of mutation process. If the individual has repeated IDs of nodes, we use the method as shown in Figure 4.9 to solve.

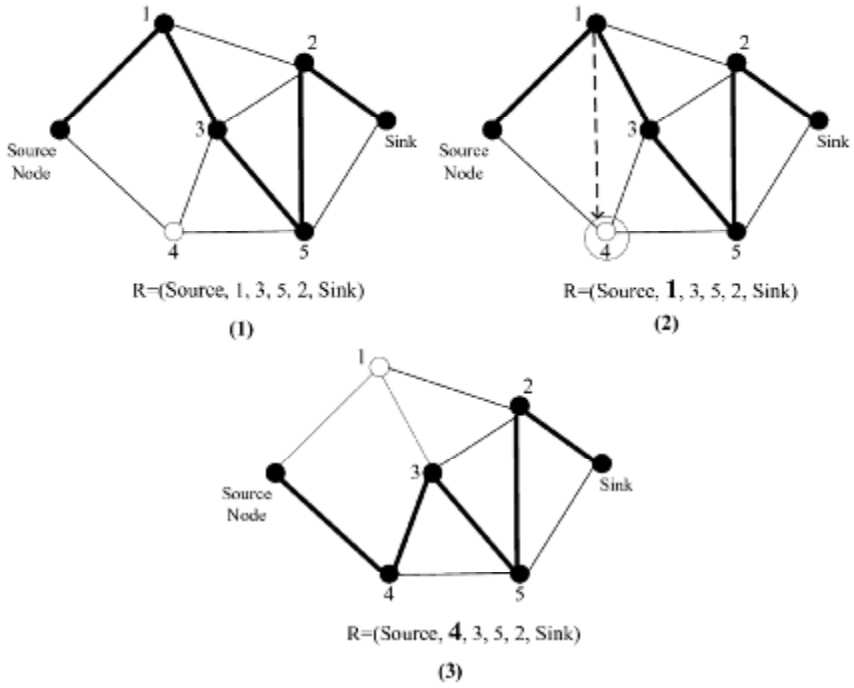

Figure 4.10: An example of the mutation operation

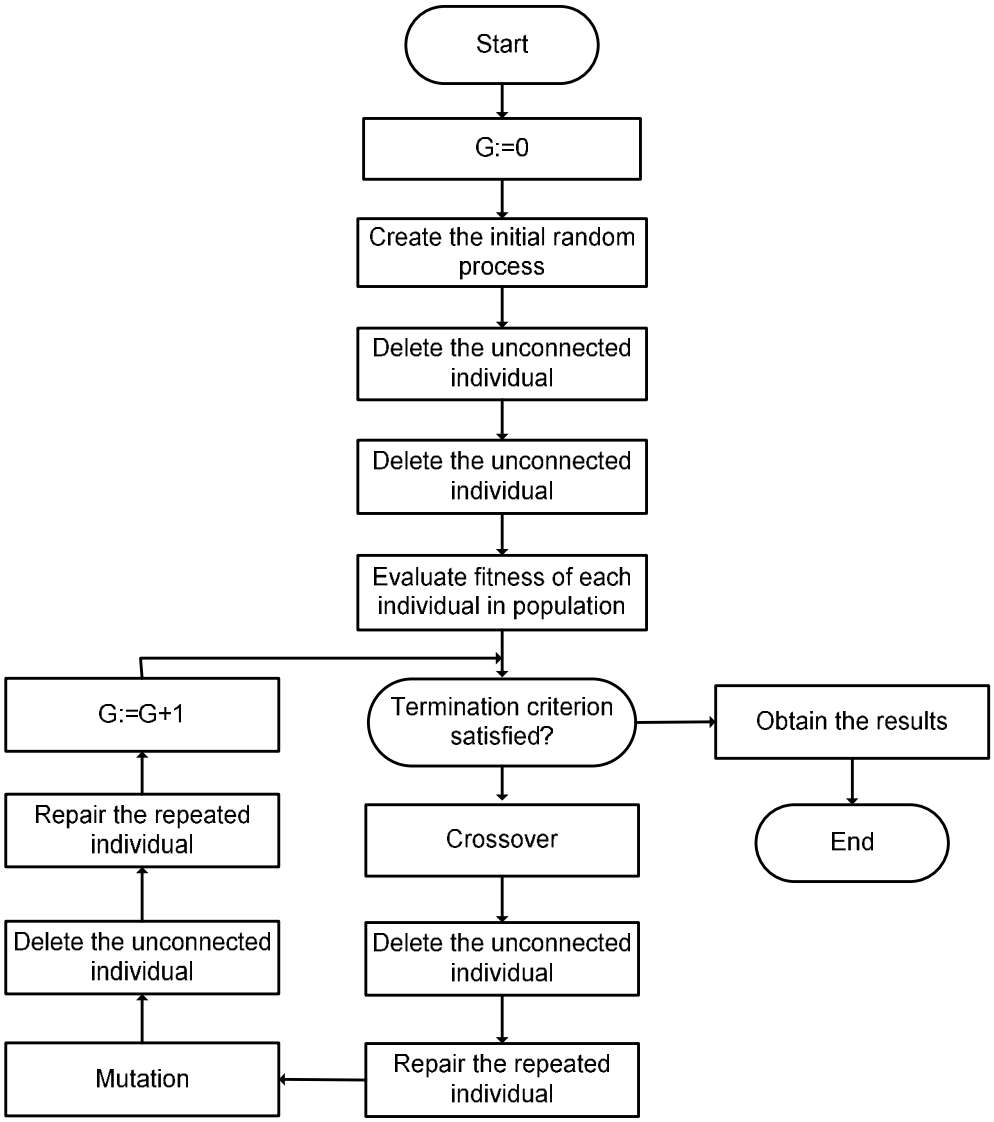

Figure 4.11: The flowchart of the proposed centralized algorithm

Having generated a new population, it is necessary to decide which of them are fittest in the sense of producing the best solutions to CPRP. To do this, a fitness function is required to provide a measure of the suitability of the solutions. The fitness function is equal to the sensing coverage of routing path which is described in Eq.4.1 and can be evaluated by the values of weights of the edges in the routing path. The detailed method of computing the weight has been presented in the previous part. A chromosome having the maximum fitness value among a population is called

elite and carried through unchanged to the next generation. Figure 4.11 shows the procedure of proposed centralized algorithm.

4.5. Results and Discussion

Extensive Simulations have been carried out to test the effectiveness of the proposed algorithm. The upper bound of the sensing coverage of routing is defined as $N_p \times (K + 1)$. A total of 100 sensor nodes with the same sensing range of R_s=7 are randomly distributed within a square area measuring 50×50 units. The source and sink nodes are located at the bottom left and upper right of the area, respectively. We further require that the total number of hops from source to sink should be no more than 7 (i.e. K=7).

First, we assume that when calculating the sensing coverage of routing, using 2000 points uniformly distributed in each circle is able to give accurate enough estimation (further discussion of this assumption will be given later). Figure 4.12(a) illustrates the path selected by the proposed algorithm in a typical simulation. It can be seen that the selected path satisfies the hop count requirement and the total sensing area covered by the nodes in the path is 15784, which is very close to the theoretical upper bound 16000. By comparison, a typical path with length of 7 is shown in Figure 4.12(b). The total sensing area is however only 13825, about 87% of the first one.

For a better comparison, we use brute force method to search all 7-hop paths from source to sink in the graph and calculate the average sensing area of all the obtained paths. The simulation is repeated 20 times. In each time, the deployment of 100 sensors with R_s=7 are randomly generated. The sensing area of the path obtained by the proposed distributed algorithm, centralized algorithm and the average sensing area of the 7-hop paths in each simulation are illustrated in Figure 4.13. It can be

seen that on average, the sensing area of the path by the proposed distributed is the same as the centralized algorithm. And the sensing area of the path by the proposed algorithm is about 1.3 times of the sensing area when only hop constraint is considered.

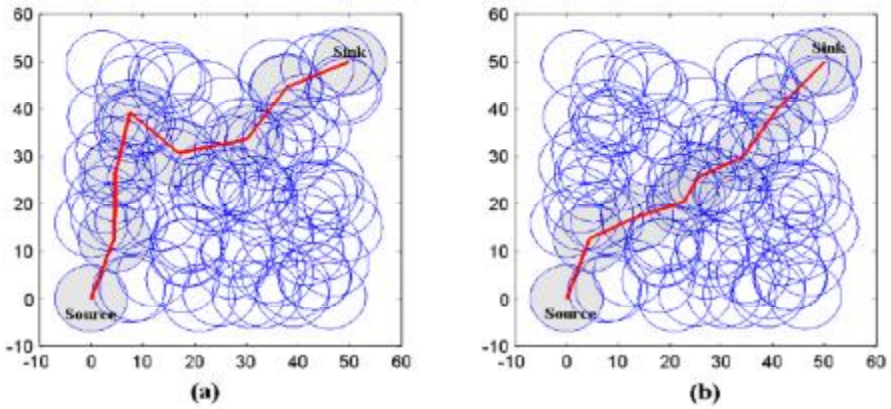

Figure 4.12: The path selected by the algorithm

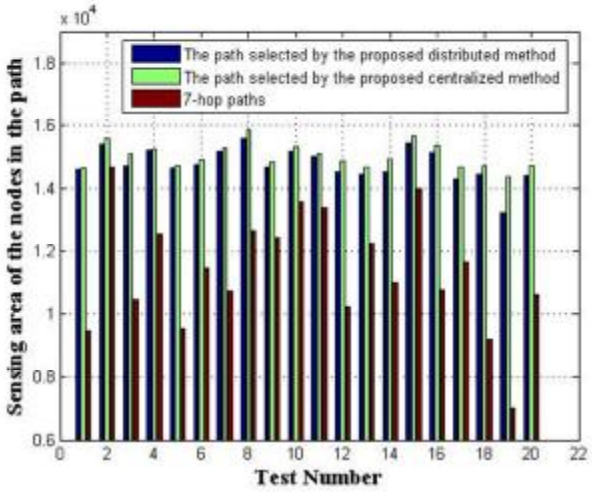

Figure 4.13: The sensing areas of the path obtained by the proposed methods and the 7-hop paths

It is also of interest to see the effect of the hop count constraint. When the deployment of sensors is fixed, we increase the hop count constraint from 7 to 14 and for each hop count constraint; a total of 20 simulations is implemented. Figure 4.14and Figure 4.15illustrate the average sensing area of 20 simulations for each hop count constraint. In Figure 4.14, the ratio of the sensing area of the path obtained by the proposed distributed algorithm to that by centralized algorithm floats around 1:1.1. And in Figure 4.15, it is interesting to find that with the increase of hop count constraint, the benefit of using the proposed method to select path is more obvious.

Finally, we will discuss that when calculating the sensing area, how many points which are uniformly distributed in each circle are able to give accurate enough estimation. This is an interesting problem since too sparse points in a sensing disk will cause undesirable effect when choosing the path while excessively increasing the number of points can significantly increase the computation time and may also easily exceeds the memory available in each sensor node. To find the answer, we increase the value from 5 to 2000, and then observe the change of the sensing coverage of the routing and the computation time of proposed distributed algorithm. The simulation is implemented 20 times and the average results are shown in Figure 4.16. In each simulation, once the path is selected given the number of points, the sensing coverage of the routing is calculated by Eq.4.6, and sensing disk of each sensor node contains 2000 points. It is interesting to find that when the number of points is larger than 50, the selected path is stabilized. The results indicate that 50 can give as accurate results as 2000 in this simulation.

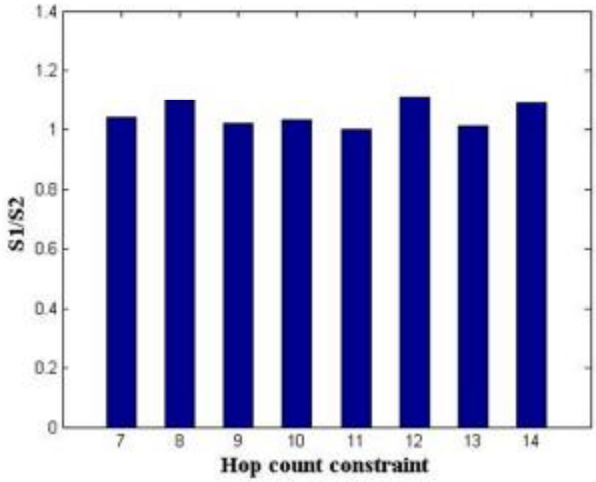

Figure 4.14: S1/S2, S1: the sensing areas of the path obtained by the proposed centralized method, S2: the sensing areas of the path obtained by the proposed distributed method

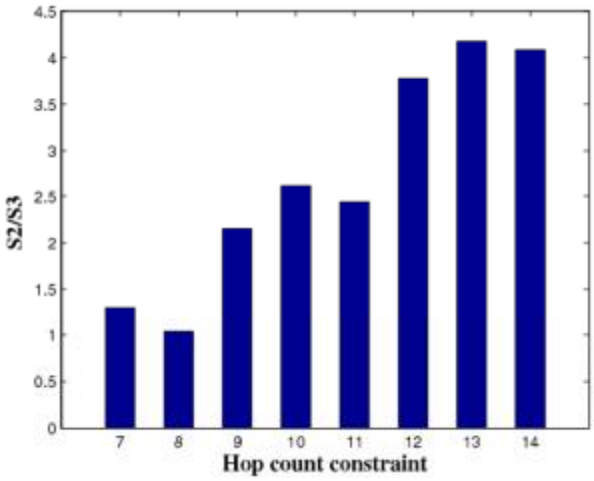

Figure 4.15: S2/S3, S2: the sensing areas of the path obtained by the proposed method, S3: the average sensing areas of the k-hop paths

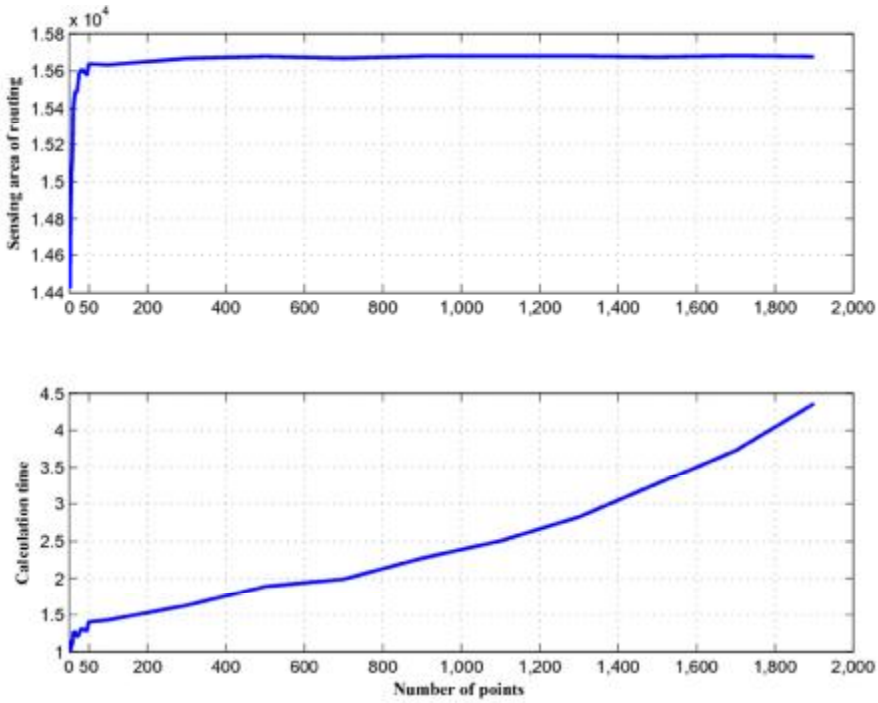

Figure 4.16: The effect of the number of points on the obtained sensing coverage (upper) and computational time

4.6. Extension to Irregular Sensing Areas

In realistic situations, the sensing area of a sensor is not necessarily a circle. In most cases, it is likely irregular which is as shown in Figure 4.17. Fortunately, our algorithms can be applied to irregular sensing areas with few modifications.

Given the sensing region of sensors are irregular and the area of the sensing region $A_{ir}^{v_i}$ of sensor v_i is known. It remains a problem how to calculate the overlapping area among the sensors. The method proposed to calculate the sensing coverage can be used which is illustrated in Figure 4.18. Then based on the obtained matrix, according to the list of neighbors, the proposed distributed or centralized algorithm is applied to find the routing path with maximum sensing coverage.

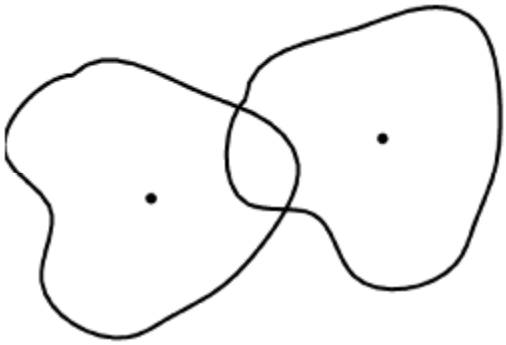

Figure 4.17: The CPRP with irregular sensing areas

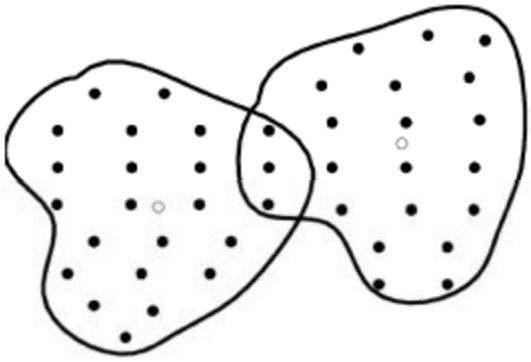

Figure 4.18: Calculating the irregular sensing areas

4.7. Summary

In this chapter, we have proposed the distributed and centralized coverage-preserving routing algorithm for randomly distributed WSNs. This work is modified from Monte-Carlo integration method, LS algorithm and GA. The modified Monte-Carlo integration method is used for obtaining the overlapping matrix of the neighboring nodes and computing the sensing coverage. Based on the matrix, in distributed version, the modified LS algorithm is used for finding the routing with maximum sensing coverage and the K hops constraint. For centralized version, the

algorithm based on GA is proposed. According to simulation results, it is found that compared with the k-hops path, the proposed algorithms can find the routing with better sensing coverage. The sensing coverage of routing by the proposed algorithms is about 1.3 times of the sensing coverage obtained by the *k*-hops path. We also analyzed the relationship between the number of points and the sensing coverage of routing. The upper bound of number of points is found in the analysis.

Chapter 5. Construction of Dynamic Data Aggregation Tree in Wireless Sensor Networks-based Object Tracking

In this chapter, we introduce the proposed localized selection scheme and the construction method of data aggregation tree for object tracking in WSNs. In this chapter, firstly, Section 5.1 overviews this work. The preliminary models used in the paper are described in section 5.2. In section 5.3, we first propose how to select the next sensor set to be woken up. Then the distributed algorithm for constructing the dynamic data aggregation tree is proposed. The simulation and discussion of the proposed algorithm are presented in section 5.4. Finally, section 5.5 concludes the paper.

5.1. Overview

Object tracking also known as target tracking, is one of the most important applications of WSNs. Generally, object tracking in WSNs is defined as tracking of targets ranging from security attacks [HK09], to moving objects in civil surveillance, and to changes in temperature [NS08] and acoustics [CH03] in environmental monitoring. However, currently, most research focus on tracking the location of moving objects. The basic operations of sensors in object tracking include (1) the generation of track information continuously; (2) the delivery to a collector/sink in real-time. For the first operation, the algorithms for evaluation of the location of the target and selection of next sensor set are designed. Kalman Filters [WB01] and Particle Filters [GS93] are the representative methods for evaluating the state of target and filtering out the noise. For saving energy, when the target is detected, only a set of sensors around the target needs to be woken up. For tracking the target continuously, the current active sensors wake up a next set of sensors to track the

target based on the possible location at next time slot predicted by the sensors/data centre. Entropy-based solutions [RE07] and the information-driven solution [AL08] are the common methods for set selection. As to the second operation, the algorithms of information aggregation usually include two types: tree-based [ZC04][ZC+04] [HI04] and cluster-based [HD03] [MP08].

Although sensor set selection and information aggregation has been studied, there is a lack of research combining these two issues together. In [ZC04], the dynamic convey tree is constructed when the target is moving. However, in the paper, the network is divided into grids and each cluster head in the grid always keeps active to track the target. The method that consumes energy and the lifetime of the network is short. In [LCZ10], the wake up scheduling is designed based on the pheromone of each sensor, however, the routing from source nodes to sink node is not discussed. In the object tracking systems [HK06][TL09], though two issues are considered together, data aggregation tree is constructed by the centralized algorithm.

In this paper, we propose the algorithm which consists of two components: 1) sensor set selection based on the prediction model and 2) dynamic aggregation tree construction based on the energy consumption model. The design of the algorithms is by no means a trivial task and needs to address two challenges. The first challenge is how to find the next sensor set to be woken up in localized way. The second challenge is how to select the cluster head when the target is moving. The third challenge is how to find the path from the cluster head to sink node in distributed way and in real-time. For solving these issues, we design a method based on the geometry method to select the next sensor, adopt the modified LS algorithm to establish the routing path dynamically in a distributed way and the remedy rules for wake up scheduling is also proposed. Compared with the existing works, we have the following contributions:

- We propose a localized algorithm that the next sensor is woken up by the active sensor independently.

- We propose an algorithm to maximize the residual energy of the tree with minimized number of nodes from the source nodes to the sink node.

- The proposed algorithm is fully distributed and does not need the location information.

5.2. Preliminary Models

In this section, we introduce assumptions and preliminary models.

5.2.1. Sensing and Communication Model

In this paper, the sensing area of the sensor in the network is assumed as the disk shape. The sensing range of the sensor is R_s. Within the sensing range R_s, a sensor node N_i can measure its distance D_i and orientation θ to a target O. We assume the sensing model (5.1), which is practical in the real environment [ZC07]. The sensing models are based on radio signal propagation models in which signal strength decays as a power of the distance.

$$P(D_i) = \begin{cases} 1, D_i \leq R_1 \\ e^{-\lambda(D_i - R_1)^\gamma} R_{max} > D_i > R_1 \\ 0, D_i > R_1 \end{cases} \quad \textbf{(5.1)}$$

where, R_1 is the initial uncertainty in detection of sensor and the parameters λ and γ are adjusted in terms of the physical conditions of the sensor. R_{max} is sensing range of the sensor with maximum value. Therefore, if the node is nearer to the object, the detection probability is higher.

The communication range of the sensor is R_c should be at least twice the sensing range R_s, which can guarantee the network connectivity. We use the Boolean communication model which is given by Equation (5.2) in this paper.

$$P(x) = \begin{cases} 1, x \leq R_c \\ 0, x > R_c \end{cases} \quad \textbf{(5.2)}$$

where, x is the distance between two nodes which can estimated by the radio. When the distance x is smaller than R_c, the two nodes can communicate with each other.

5.2.2. Target Motion Model

The motion mode of target can be classified as two classes.

(1) Random motion mode. In this class, the target goes through the area randomly. The trajectory of target is estimated by the sensed data from sensor and previous sensed data [TS08].

(2) Predefined motion mode. The object goes through the area based on a predefined motion mode. The CT and the CV mobility model are the usual mobility models that are used in object tracking [TB09]. The uniform CA is used in [TB09]. The motion model can also be a directional motion model [KA08] that movement models take into account the moving direction to imitate the actual object motion.

In this paper, the predefined target motion model is used. We assume that the target goes through the network with a constant speed. When the target change the direction of the motion, the location of the target is still needed to be tracked continuously. In the assumption of the paper, we use CT model to describe the motion of the target. The model assumes that the target changes the moving direction

with constant speed and the turn rate is predefined as θ. We use CT model and CV models to describe the target motion as follows [9]:

$$X_k = A_k X_{k-1} + Q_k \mu_k \quad \textbf{(5.3)}$$

where $X_k = [x, y, v_x, v_y, \theta]^T$ is the target state at the time k, x, and y are the target position coordinates. v_x and v_y are the corresponding velocities of the x and y coordinates. $\mu_k \sim N(0, g_k)$ is the process noise; A_k denotes the state transition matrix. A_k and Q_k can be described as

$$A_k(CT) = \begin{bmatrix} 1 & 0 & sin(\theta T)/\theta & (cos(\theta T)-1)/\theta & 0 \\ 0 & 0 & cos(\theta T) & -sin(\theta T) & 0 \\ 0 & 1 & (1-cos(\theta T))/\theta & sin(\theta T)/\theta & 0 \\ 0 & 0 & sin(\theta T) & cos(\theta T) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(5.4)}$$

$$A_k(CV) = \begin{bmatrix} 1 & 0 & T & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & T & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \textbf{(5.5)}$$

$$G_k(CT) = G_k(CV) = \begin{bmatrix} 1/2T^2 & 0 & 0 \\ T & 0 & 0 \\ 0 & 1/2T^2 & 0 \\ 0 & T & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{(5.6)}$$

where $A_k(CV)$ means the state transformation equation in CV model; $A_k(CT)$ means the state transformation equation in CT model.

5.2.3. Prediction Model

KF is a set of mathematical equations that provides an efficient computational solution to discrete time data filtering problems, in essence removing extraneous noise from a given stream of data. A KF has two distinct phases: predict and update. The predict phase uses the state (e.g. location) estimate from the previous timestamp to produce an estimate of the state at the current timestamp. In the update phase, measurement information at the current timestamp is used to refine this prediction to arrive at a new, (hopefully) more accurate state estimate, again for the current timestamp.

In this paper, based on the predefined motion model, the Distributed KF (DKF) which consists of low-pass consensus filter and micro KF on each node, is used for locally estimate the state of target. The aim of a distributed filter [HR10] is to parallelize the state estimation such that every node determines its state estimate independently. Low-pass consensus filter and micro KF are described as follows [S05]:

The Kalman filter uses the linear stochastic difference equation to estimate the state $x \in \Re^n$ of a discrete-time controlled process

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \qquad \textbf{(5.7)}$$

with a measurement $z \in \Re^m$ that is

$$z_k = Hx_k + v_k \qquad \textbf{(5.8)}$$

We assume v_k and w_k are independent of each other, white, and with normal probability distributions. They are the random variables which note the measurement noise and the process separately.

$$p(w) \sim N(0, Q) \quad \textbf{(5.9)}$$

$$p(v) \sim N(0, R) \quad \textbf{(5.10)}$$

Q and R represent the process noise covariance and measurement noise covariance respectively. In this paper, we assume they are constant, however, in real environment, Q and R might change with each time step or measurement.

(5.11) is difference Equation. The matrix A $(n \times n)$ in (5.11) concerns the state at the time step k-1 to the state at the time step k, when there is not a driving function or process noise. A is constant in this paper, but it might change in real world. The matrix B $(n \times 1)$ is correlation matrix of the optional control input $z \in \Re^l$ to the state x. The matrix H $(m \times n)$ is correlation matrix that relates the state to the measurement z_k. H is constant here but might change in practice.

$$\dot{s}_i = \sum_{j \in N_i} (s_j - s_i) + \sum_{j \in N_i \cup \{i\}} (u_j - s_i) \quad \textbf{(5.11)}$$

where s_i is the state of node i in m-dimensional, u_i is the input of node i in m-dimensional. The (5.11) can be equal to

$$\dot{s} = -\hat{L}s - \hat{L}u + (I_n + \hat{A})(u - x) \quad \textbf{(5.12)}$$

where $s = col(s_1, \ldots, s_n)$, $\hat{A} = A \otimes I_m$ and $\hat{L} = L \otimes I_m$ notes a low-pass consensus filter with Equation (5.13) from input u to input x. (5.13) is the MIMO transfer function that is as follows

$$H_{lp}(v) = \left[(v+1)I_n + \hat{A} + \hat{L}\right]^{-1}(I_n + \hat{A}) \quad \textbf{(5.13)}$$

This filter is to fuse the measurements. It applies the algorithm to $H_i' R_i^{-1} z_i$ to calculate $\hat{y}_i$ as the input of node i.

5.2.4. Energy Consumption Model

In relation to our model, we defines the total energy consumption as $E_{ij}(k)$ that means a packet with k bit is transmitted from a node i to a node j. $E_{ij}(k)$ is described as follows:

$$E_{ij}(k) = ak + bkd_{ij}^2, \text{ if } j \text{ is sink node} \quad \textbf{(5.14)}$$

$$= (a + c)k + bkd_{ij}^2, \text{ otherwise} \quad \textbf{(5.15)}$$

We should note that it is different when one packet is transmitted to the normal node and to the sink node. Because the sink node has unlimited energy, we will ignore the cost for receiving messages.

5.3. Problem Definition

It is assumed that there are N sensors randomly deployed in a given area. Each sensor has the identical sensing range R_s and communication range $R_c(R_c \geq 2R_s)$. The sensing area of node is regarded as the circle. The location of the sensors are unknown, but the distance $d(i,j)$ of each neighbor i, j can be estimated by radio. The directional information (the orientation of each neighbor) $\theta_{v_i v_j}$ is also a prior knowledge which can be obtained by the antenna. $\theta_{v_i v_j}$ is the angle between node $v_i$and node v_j. The remaining energy of each node is En_i, and the energy is consumed according to the previous energy consumption model. A object goes through the area according to the above mentioned motion model. If the sensor detects the object at time instant k, the sensor sends the sensing data to the sink and wakes up a set of nodes which will be active at time k+1. The objective of the problem is to use the minimum number of nodes to construct a data aggregation tree while satisfying the constraint that the minimum remaining energy of the non-source node v_h in the tree should be maximized. At the same time, the residual energy of the non-source node

in the tree should be greater than the predefined threshold Th_{En}. The problem formulation is:

Let G be an undirected graph $G=(V,E)$, where $V=\{v_0,...,v_N,t\}$ is the set of nodes, $|V|=N+1$,E is the set of edges, t is a specified node. If $d(i,j) \leq R_c$, there is an edge between the node v_i and v_j. The sensing range of each node is R_s and the communication range of each node is R_c. The remaining energy of each node is En_i. The objective is to construct a data aggregation tree T, where the root is t, a subset of $TS_{sou}=\{v_g,...,v_h\}$ is the set of source nodes which is woken up to detect the object, so that

$$min \sum_{e \in T} W(e) \qquad W(e) = 1 \qquad \textbf{(5.16)}$$

$$\text{subject to} \quad max\, min\, En_f \; f \in \{T - S_{sou} - t\} \qquad \textbf{(5.17)}$$

$$\forall f \in \{T - S_{sou} - t\}: En_f > Th_{En} \qquad \textbf{(5.18)}$$

5.4. A Distributed Data Aggregation Method for Object Tracking

In this section, we describe the distributed data aggregation method which is executed on each node to collect the data and send the data to sink for object tracking. The algorithm is executed when the location of the node is predicted by the sensors. According to the possible next location of target which has been predicted on the sensors, a next set of active nodes is woken up and the data aggregation tree for the next set is also constructed.

5.4.1. Basic Idea

We will explain the basic idea of the proposed algorithm before describing the method in detail. When the target is detected by a set of active sensors at time k, the sensors use Distributed KF to estimate the current location of the target and predict the possible location of the target at the next time k+1. According to the next location of the target, the current active sensors wake up the next set of sensors which are around the predicted location of the target by using the geometry method. According to the current data aggregation tree and next set of active sensors, a new data aggregation tree is constructed by the proposed heuristic algorithm. The time of the construction should be minimized and the number of sensor nodes of the tree should also be minimized.

5.4.2. Wakeup Control Method for Object Tracking

Assume that the active sensor Se_i knows the location $L_k=(x_k,y_k)$ of the target at the time step k and the predicted location $L_{k+1}=(x_{k+1},y_{k+1})$ of the target at the time step k+1. The active sensor can also detect the orientation of the target which is expressed as δ_i. The direction angle of each neighbour $\theta_{Se_iSn_j}$ is a prior knowledge. The distance between the active sensor and the neighbour is $d(Se_i,Sn_j)$. As shown in Figure 5.1, according to the cosine law, the distance $d(Sn_i,L_{k+1})$ between the neighbour node and the target at the next time step k+1 and the distance between the target and the node is $d(Se_i,L_{k+1})$ can be obtained as follows:

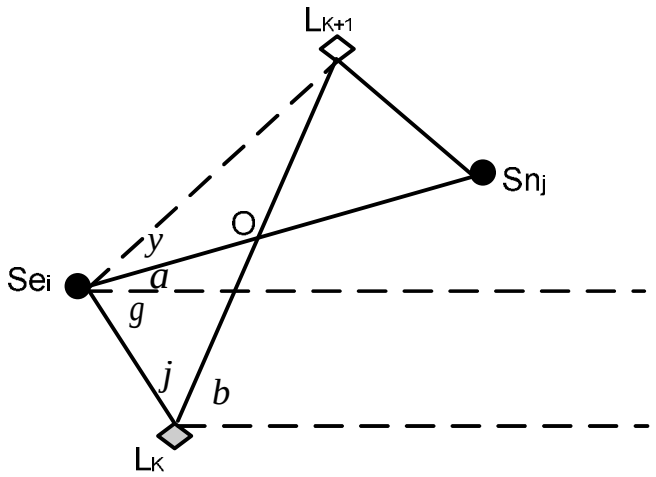

Figure 5.1: The location and angles of the target and sensors at different time instants

$$d(Se_i, L_{k+1}) = \sqrt{d(Se_i, L_k)^2 + d(L_k, L_{k+1})^2} - 2 \times d(Se_i, L_k) \times d(L_k, L_{k+1}) \times cos\,\varphi \quad \textbf{(5.19)}$$

where $d(L_k,L_{k+1})$ can be obtained through the Euclidean distance equation

$$d(L_k, L_{k+1}) = \sqrt{(x_k - x_{k+1})^2 + (y_k - y_{k+1})^2} \quad \textbf{(5.20)}$$

φ is equal to

$$\varphi = 180° - \gamma - \beta \quad \textbf{(5.21)}$$

If $d(Se_i,L_{k+1}) > R_s$, it means that the sensor Se_i cannot detect the target at time instant $k+1$. If $d(Se_i,L_{k+1}) < R_s$, based on the $d(Se_i,L_{k+1})$, the $d(Sn_i,L_{k+1})$ can be obtained as

$$d(Sn_i, L_{k+1}) = \sqrt{d(Se_i, L_{k+1})^2 + d(Se_i, Sn_i)^2} - 2 \times d(Se_i, L_{k+1}) \times d(Se_i, Sn_i) \times cos\,\psi \quad \textbf{(5.22)}$$

Since $\angle Sn_i Sc_i L_k$, $\angle L_{k+1} L_k Se_i$ and $d(Se_i,L_k)$ are known, as shown in Figure 5.2, $d(Se_i,O)$ and $d(L_k,O)$ are equally expressed as

$$d(Se_i, O) = \frac{d(Se_i,L_k)\tan\varphi}{[\tan(\alpha+\gamma)+\tan\varphi]\cos(\alpha+\gamma)} \quad \textbf{(5.23)}$$

$$d(L_k, O) = \frac{d(Se_i,L_k)\tan(\alpha+\gamma)}{[\tan(\alpha+\gamma)+\tan\varphi]\cos\varphi} \quad \textbf{(5.24)}$$

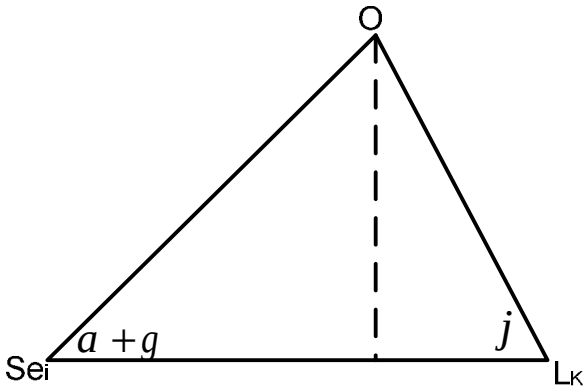

Figure 5.2: The location of sensor Se_i and the target at time k

As shown in Figure 5.3, in $\Delta Se_i O L_{k+1}$, three edges are known, according to the equation as

$$\cos\psi = d(Se_i, O)^2 + d(Se_i, L_{k+1})^2 - d(L_{k+1}, O)^2/2 \times d(Se_i, O) \times d(Se_i, L_{k+1})$$ **(5.25)**

Thus $d(Sn_i,L_{k+1})$ can be obtained based on cosine theory,

$$d(Sn_i, L_{k+1}) = \sqrt{d(Se_i, L_{k+1})^2 + d(Se_i, Sn_i)^2} - 2 \times d(Se_i, L_{k+1}) \times d(Se_i, Sn_i) \times \cos\psi \quad \textbf{(5.26)}$$

Se_i selects a neighbour node Sn_i with the smallest value of distance $d(Sn_i,L_{k+1})$ as the next active node to wake up, since the nearer distance between the target and node means more accurate detection of the target according to the sensing model mentioned in section 2.

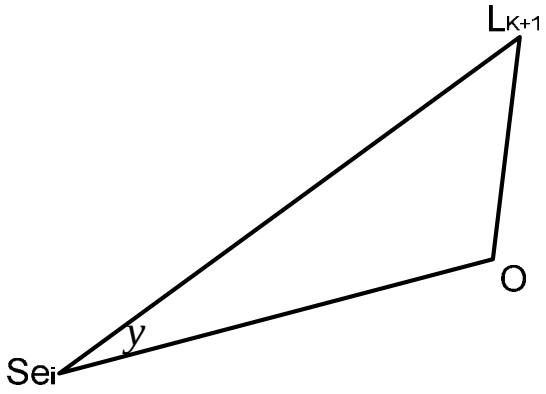

Figure 5.3: The location of sensor Se_i and the target at time $k+1$

5.4.3. Construction of Data Aggregation Tree

Since the nodes to be activated at time k+1 are selected, these nodes are regarded as the source nodes and the sink node is regarded as the root. There are two cases for construction of data aggregation tree. For the first case, when there is not repeated source node at time k+1 compared with the source nodes in data aggregation tree at time k, the tree needs to be reconstructed completely. The method for the first case is called as reconstruction of data aggregation tree. The objective of the method is to minimize the number of middle nodes in the tree and maximize the residual energy of middle nodes. The second case is that when there are repeated source nodes, the new tree is constructed by pruning or adding the branch of tree at time k. The method for the second case is called as dynamic construction of data aggregation tree. The objective of the method is that the time of the reconfiguration should be minimized

and the number of sensor nodes in the tree should also be minimized at the same time.

1) Reconstruction of Data Aggregation Tree. Assume there is a set of nodes N_k=(n_1,n_2,…, n_i) needs to be woken up at time k+1. And the set of active nodes N_{k+1}=(c_1,c_2,…,c_j) can detect the target at time k. If $\emptyset = N_n \cap N_c$, the reconstruction of data aggregation tree is needed. Since the source nodes can detect the target, they can communicate with each other. There are two steps of the method: (1) elect a cluster head from the set of nodes N_k, N_k is regarded as the set of source nodes; (2) find a routing from the cluster head to the sink node.

The distance between the possible location of the target at time k+1 and the source node is $d(T_{k+1},n_l)$, therefore, the weight W_{n_l} of the source node is equal to:

$$W_{n_l} = En_{nl} / d(T_{k+1}, n_l) \qquad \textbf{(5.27)}$$

We select a maximum value ofW_{n_m} in the source nodes as the cluster head. When the cluster head is selected, the other source nodes will communicate with the cluster head directly. Then the label setting algorithm is modified to find the routing from cluster head to the sink node. The detailed algorithm is presented as follows.

Let each node has a set of labels in which each label relates to a different path from the sink t to that cluster head. And each label has two numbers by denoted as $(H_{v_i}^g, E_{v_i}^g)$, where $H_{v_i}^g$ and $E_{v_i}^g$ represents the length of g routing path and the minimum value of residual energy in the g routing path respectively. The $H_{v_i}^g$ and $E_{v_i}^g$ of the node v_i can be computed as $H_{v_i}^g = H_{pre_{v_i}}^{g+1} + 1$, $E_{v_i}^g = \min\{E_{pre_{v_i}}^g, En_{v_i}\}, En_{v_i}$, pre_{v_i} is the previous node of node v_i in the g path.

Definition 1. Let $(H_{v_i}^g, E_{v_i}^g)$ and $(H_{v_i}^q, E_{v_i}^q)$ be two labels related to two different path g q at node v_i. We say that $(H_{v_i}^g, E_{v_i}^g)$ is smaller than $(H_{v_i}^q, E_{v_i}^q)$ if and only if $(H_{v_i}^g > H_{v_i}^q)$.

Definition 2. A label is said to be *possible* if the label is not smaller than any other label at the node.

Definition 3. A path g is said to be *feasible* if the corresponding label is possible at each node of the path.

Definition 4. A path g is said to be *best* if it is feasible and $E_{v_i}^g$ is greater than the weight of any label on the nodes.

The proposed algorithm is to find the possible label on each node and the best path from sink to cluster head. It has two steps: (1) Initializing the labels when the sink broadcast the query to the network; (2) Updating the labels and finding the routing from the source node to the sink.

Ⅰ Initializing the Labels

At first, there is no label on any node except for the label $(0, \infty)$ on sink t. ∞ means the sink has the infinite energy. Sink sends the query to the neighbors at the start. The node v_i receiving the query adds the label $(H_{v_i}^g, E_{v_i}^g)$ to $Label_{v_i}$, where $H_{v_i}^g = 1$, $E_{v_i}^g = En_{v_i}, g \in I_{v_i}, I_{v_i}$ is the index set of labels, $Label_{v_i}$ is a set of labels on node v_i. Then the neighbors broadcast the message with the label and the query. The node v_j receiving the message processes the received label $(H_{v_i}^g, E_{v_i}^g)$, $E_{pre_{v_j}}^g = E_{pre_{v_j}}^g - E_{sd}$, $H_{v_j}^g = H_{pre_{v_j}}^g + 1$, $E_{v_i}^g = \min\{E_{pre_{v_i}}^g, En_{v_i}\}$, E_{sd} is the energy consumed when the pre_{v_i} sends the query to the node v_j. And the received label after processing is compared with the stored labels in $Label_{v_j}$. If one of label in $Label_{v_j}$ is smaller than the received label, it should be dropped. Otherwise, it is a possible label that will be

stored in $Label_{v_j}$. The operation will repeat until each node receives all the possible labels. Finally, the best path Pb to the sink node is obtained at the each node. The $H_{v_i}^{Pb}$ which is the minimum hop counts from the node v_i to the sink node is regarded as the level Le_{v_i} of the node v_i.

Updating the Labels and Finding the Routing

Since the residual energy will be consumed after the operations of the transmission and the sensing are executed, En_{v_i} is a dynamic value. When the residual energy of the node is smaller than the predefined threshold Th_{En}, the node is called dangerous node. The dangerous node broadcasts the alarm information to the neigboring nodes. The neigboring node receiving the information will check its label. If the sending node is the previous node in one of the path of the labels, the corresponding label will be dropped. Then the information will be forwarded to the neigboring nodes of the receiving node. If the sending node is not in the path of the labels, the information will be dropped. The information will be broadcasted until all the related nodes receive the information and update the labels. The related node means the dangerous node is in the shortest path from the node to the sink.

When the possible location of the target is predicted, a set of the nodes becomes active at time k+1. According to the updated value of the residual energy of the active nodes, the weight of the node is computed. Since the label will be updated as the energy is consumed, the election method of the cluster head will be changed as follows:

1) The source node with the highest value of weight is select as a candidate of the cluster head.

2) If $Label_{CH_{k+1}} = \emptyset$, it means there exists the dangerous nodes on each path in the $Label_{CH_{k+1}}$. In this case, we need to select the other source node as the cluster head. Therefore, the node v_m with the weight $W_{CH_{k+1}} \geq W_{v_m} \geq W_{v_n}, \forall v_n \epsilon N_c, N_c = N_{k+1}, N_c = N_c - CH_{k+1}$ is selected as the cluster head candidate. The cluster head candidate will be noted as CH_{k+1}.

3) The step 2 will repeat until $Label_{CH_{k+1}} \neq \emptyset$. The routing is built based on the $Pb \in Label_{CH_{k+1}}$.

4) If $Label_{v_n} = \emptyset, \forall v_n \epsilon N_c, N_c = N_{k+1}$, a random neighoring node *Ne* of the source node with highest weight will be selected.

5)If $Label_{Ne} = \emptyset$, the node v_m with the weight$W_{CH_{k+1}} \geq W_{v_m} \geq W_{v_n}, \forall v_n \epsilon N_c, N_c = N_{k+1}, N_c = N_c - CH_{k+1}$ is selected as the cluster head candidate. The cluster head candidate will be noted as CH_{k+1}.A random neighoring node of the source node $CH_{k+1}$with highest weight is *Ne*.

6) The step 5) will repeat until $Label_{Ne} \neq \emptyset$. The source node is regarded as CH_{k+1}. The routing from clusterhead to the sink is (CH_{k+1}, Pb_{Ne}).

After the cluster head is selected, since there are labels on each node, the routing from the cluster head to the sink node is also determined. The updated residual energy of the nodes on the routing will be sent to the next node of the routing integathering with the sensed data. Therefore, the $E_{v_i}^g$ of the related label in the node v_i will be updated by comparing the $E_{v_j}^g$ of sending node v_j with current value of residual energy of v_i. The minimum value of the residual energy is regarded as the $E_{v_i}^g$. According to the method, the labels of the all nodes on the routing are updated.

However, there exists a case as follows. When the sensed data is sent back along the routing, there may be the node in which $Label_{v_n} = \emptyset$. It means that the node v_n has

received the alarm information or $En_{v_n} < Th_E$, and all the labels of the node are dropped. Furthermore, the node v_n receives the sensed data message before broadcasting the alarm information to the neighbours. Thus, when the node received the sensed data message, it replied the alarm information to the sending node v_m. If $|Label_{v_m}| > 2$, the related label is dropped and the node in the any one of the rest labels is selected as the next hop of the routing. If $|Label_{v_m}| = 1$, v_m select a neighbour which $|Label_{Ne}| \geq 0$, as the next node on the routing. Then the routing will be traced along the$Label_{Ne}$.

The above steps and solutions will be repeated until the sensed data is received by the sink node. We assume that the network guarantee the connectivity, thus the proposed routing method can always converge.

2) Pruning the Data Aggregation Tree. When the target moves through the network, the data aggregation tree is also changed because of difference between the sets of source nodes at different time instant. However, it consumes time and energy if reconstruction of the tree at each time instant. We found that $N_k \cap N_{k+1} \neq \emptyset$ if the distance between the location of the target at the time step k and k+1 is $d(T_k, T_{k+1}) < 2R + \varepsilon$. Therefore, we can reconfigure a new data aggregation tree at the time step k+1 by pruning the data aggregation tree at the time step k. The rules for pruning the data aggregation tree are as follows:

1) As shown in Figure 5.4, if $N_k \cap N_{k+1} = O(k, k+1)$, $CH_k \epsilon O(k, k+1)$ and CH_k is available, thus $CH_k = CH_{k+1}$ and the other source nodes at time k+1 communicate with the CH_k. The node v_i is available means $Label_{v_i} \neq \emptyset$. If the CH_k is not available, the reconstruction method of the tree is used to find a new tree.

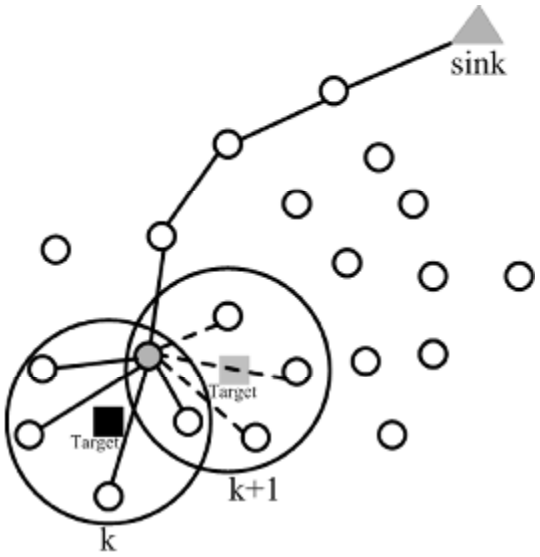

Figure 5.4: Pruning the data aggregation tree when the cluster head is in two sensor sets

2) As shown in Figure 5.5, if CH_k is not in $O(k, k+1)$, the reconstruction method of the tree is used.

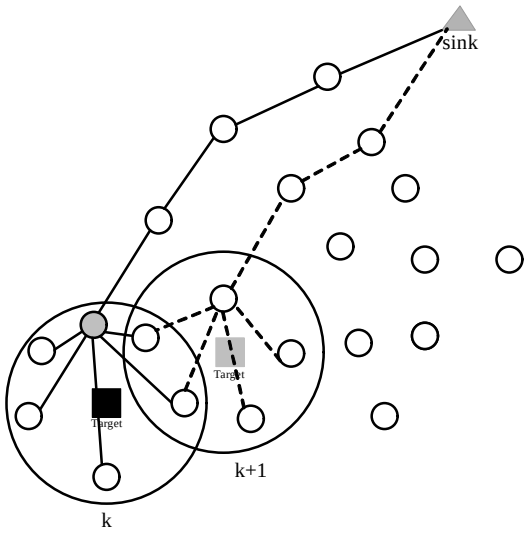

Figure 5.5: Pruning the data aggregation tree when the cluster head is not in two sensor sets

5.4.4. Remedy for Waking Up Method

If the predicted location of the target is not accurate, the active nodes cannot detect the target. Therefore, when a set N_{k+1} of the nodes is woken up at time k+1 by the set N_k of active nodes at time k, N_k will be active for t_a time slot. If the target cannot be detected by the set N_{k+1}, the cluster head of the set N_{k+1} sends a REMEDY message to the neighbour n_i in the set N_k. Then the node n_i broadcasts the message to the cluster members. All the cluster members wake up their neighbour nodes to track the target as shown in Figure 5.6. Thus, the target will not be missed even if there is error of prediction.

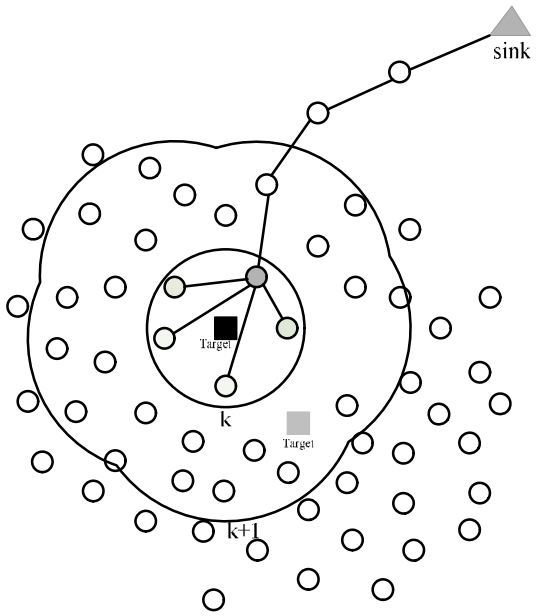

Figure 5.6: Remedy method

5.5. Simulations

At first, we will discuss the complexity of the data aggregation tree construction. We assume there are *n* nodes and *E* edges in a network is, the maximum number of data aggregation trees is *M*, the number of source nodes is *m*, in the worst case, each data

aggregation tree is needed to be reconstructed. Thus, the time complexity is $O(n^2+M*m)=O(n^2)$, the message complexity is $E^2+M*(m-1)$.

Then extensive simulations have been carried out to test the effectiveness of the proposed algorithm. A total of 100 sensor nodes with the same sensing range of R_s = 7 are randomly distributed within a square area measuring 50×50 units. The transmission range is calculated using in above model in section 5.2.1. We use a mobility model which is mentioned in section 5.2.2 to simulate the movement of a target. In this model, the target goes through the network with constant speed and the target changes the moving direction with the predefined turn rate θ. Table 5.1 lists the parameters in the simulation.

Table 5.1: Simulation parameters

Parameters	Values
size of field (m^2)	50×50
number of nodes	100
communication range (m)	14
sensing range (m)	7
the speed of a mobility target (m/s)	1.0-10.0
the turn rate (θ)	0.0-0.5
prediction accuracy (p)	0.7-1.0
size of control message (*byte*)	1.0
size of a sensing report (*byte*)	20

In this section, we compare the proposed algorithm with the DCTC method according to energy consumption, the number of active sensors involved for tracking

and the collection delay. When the deployment of sensors is fixed, we increase the speed of target from 1.0 to 10.0. For each velocity of the target, a total of 20 simulations are implemented.

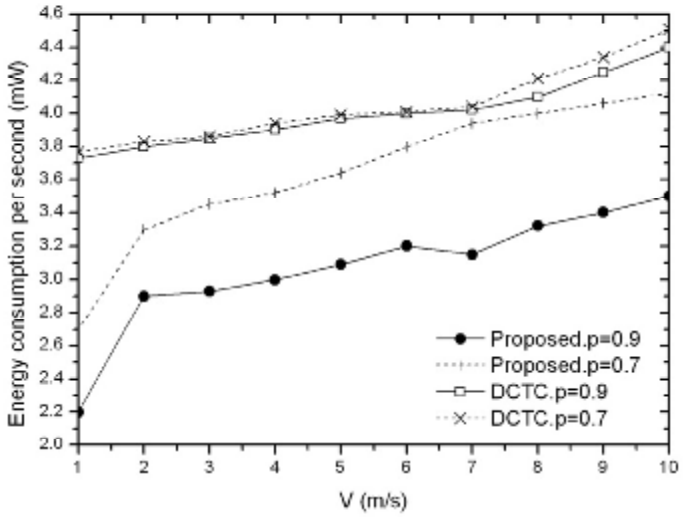

Figure 5.7: Comparing the energy cost of different algorithms

We first compare the proposed algorithm with the DCTC according to energy cost. As shown in Figure 5.7, the proposed algorithm uses less energy than the DCTC for constructing the data aggregation tree in object tracking. From the figure, we can also find that the energy cost of the proposed algorithm increase as the speed of the target increases. The reason is that as the velocity increases, more messages are needed to be exchanged for adding or pruning the nodes from the tree.

It is also interesting to compare the number of active sensors for tracking the target between the proposed algorithm and the DCTC. As shown in Figure 5.8, the number of nodes in proposed algorithm is less than that in the DCTC. From the above two figures, we find that the prediction accuracy affects the energy consumption and the

number of active sensors. Since the miss-prediction is found, the remedy method is executed which consumes the energy and more nodes are woken up.

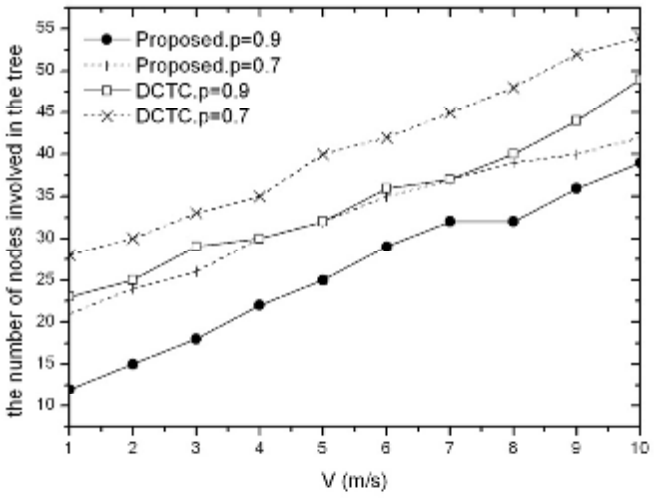

Figure 5.8: Comparing the number of nodes involved in the tree of different algorithms

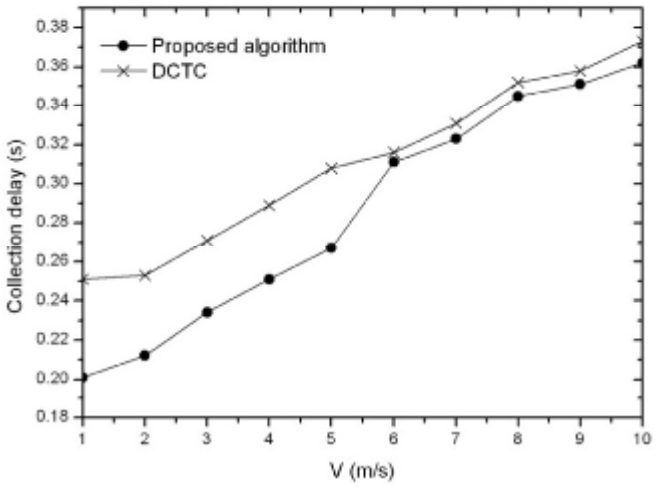

Figure 5.9: Comparing the collection delay of different algorithms

Figure 5.9 shows the collection delay when the velocity of the target increases. Collection delay in this chapter means the data is aggregated from the source nodes to the sink nodes. Since tree reconstruction is only done when the certain condition is satisfied, the tree may not be optimal when pruning the tree is executed if the moving speed of the target is high. Therefore, in the figure, it shows that the data collection delay is higher when the velocity increases. Compared with the DCTC, the proposed algorithm has shorter delay.

5.6. Demonstration of Single Object Tracking on iSensNet

We have implemented our proposed approach on our WSN-based ITS testbed, the iSensNet (Intelligent Services with Wireless Sensor Network) platform. The iSensNet platform consists of the protocols and mechanisms for MAC Layer, routing layer, and application layer, and is used to execute different protocols and application programs for demonstration and experiments. Both stationery sensor nodes and mobile nodes are installed on the test bed. The stationery sensor nodes are installed under the platform; and roadside units are installed at each intersection. Mobile nodes refer to the sensor nodes installed at each mobile model car.

In this demonstration, we want to use the light sensors that have been built under the test bed to track a car which is moving along the traffic road. We have following assumptions. 1) The location of each light sensor is predefined. 2) The ID of each light sensor is unique. 3) The light sensor is divided into two parts: communication part and sensing part. 4) The base station is connected with the computer and there is a table that stores the relationship between the ID and location. Since the location of a car at the next time slot is known according to the traffic rules, we don’t need to predict the location. The working flow is designed as follows:

Step 1: Each sensor has the sleeping schedule for the sensing part before the car shows up, e.g. sampled once every second.

Step 2: When a sensor detects the car, it sends its ID to the base station by using the proposed data aggregation tree construction method and wakes up the next sensor on the road by using the proposed wakeup control method. Then the detecting sensor sleeps.

Step 3: When the sensor receives the waking up message, it wakes up until the car pass by. And then redo step 2.

Step 4: When the base station receives the ID of the sensor, the car will be drawn on the location of the map. The result will be showed by Graphical User Interface (GUI).

We found that the proposed algorithm can be used to demonstrate streetlamp tracking. The result shows the proposed algorithm runs correctly even in the real word application. However, in this test bed, there exist some problems that are not considered in the ideal simulation. Since the streetlights are all controlled by the base station, there are two problems in the test bed. If the controlling time on the base station is greater than the time that the car passing through two streetlights, the first problem is time lag between lightening of the streetlamp and location of car. The second problem is disorder between bright light and turned off light. For solving the first problem, we use two base stations to control the streetlamp corporately that can speed up the controlling time. For solving second problem, we design a queue to store the IDs of streetlight on the sink. We can also control the light in distributed way by building a controlling sensor on each streetlamp, the sink is only used to connect with computer that shows the status of the lights and car on the map by using GUI. Thus, the above two problems can be solved.

5.7. Summary

In this chapter, we apply our studies on dynamic construction of data aggregation tree for tracking the location of the target. We first consider the sensor set selection scheme in a fully distributed way. Based on the prediction model, the current active sensors wake up the next set of sensors which are around the predicted location of the target independently. After obtaining the set of sensors to be active, the data aggregation tree from the set of sensors to the sink node is constructed quickly and energy efficiently. When the target is moving, the tree will be reconstructed in real time. Simulation results show that the proposed algorithm achieves much better performance than other method for constructing data aggregation tree, considering the number of active nodes involved, the energy consumption, and the collection delay of the tree.

Chapter 6. Conclusions and Future Works

In this chapter, we summarize our works and discuss the directions for future research.

6.1. Conclusions

Coverage, routing, and data aggregation are three important issues on the topic of object tracking in WSNs. In this thesis, we primarily investigate how to save the energy of nodes and improve the accuracy of tracking the mobile target in a WSN. The main works of the thesis can be concluded as follows.

In **Chapter 3**, considering the existing evolutionary algorithms which need global information to find the solution, we investigate the problems for which modified the evolutionary algorithm can be modified to find the solution based only on local information. In this way, the modified algorithm is suitable for solving the problem in distributed system, particularly in WSNs. We propose a localized approach for evolutionary algorithm, in which the local information is needed to find the global near optimal solution for the optimization problem in distributed system. We also analyse the impact of algorithm parameters, evaluation metrics, and some suitable applications of the proposed approach. As to the problem in WSNs, we use the proposed approach to solve the energy-efficient coverage problem. Simulations have been executed to validate the effectiveness of our localized approach for evolutionary algorithm in terms of the number of sensors needed to cover the whole area, the size of the storage in the sensors, and the calculation time.

In **Chapter 4**, considering the integration of the coverage and routing in WSNs, we define the Coverage-Preserving Routing Problem (CPRP), in which the objective is to find a routing path in a WSN with the maximum sensing coverage provided by the

nodes on the path subject to the delay constraint. We prove the CPRP is a NP-Hard problem. Then we present the preprocess that calculate the sensing area of multiple sensors. We propose two coverage-preserving routing algorithms, i.e. the Centralized Algorithm and the Distributed Algorithm, both of which use the sensing coverage between the sensors obtained by the preprocess as the weight to find a path that is subject to the hop constraint with the maximum weight. Simulation results indicate that the proposed algorithms obtain better performance than other existing works based on whole sensing coverage of the obtained path.

In **Chapter 5**, considering the data aggregation of object tracking in WSNs, we investigate the source nodes selection scheme and the construction of data aggregation tree for object tracking. We present a novel fully distributed method for sensor set selection, which can save the energy of the node and communication cost. Each active node can wake up the sensor for tracking the target based on its decision independently. The proposed selection scheme makes use of the information about the prediction location of the target and the geometry information between the neighbours. Then we propose a dynamic construction method for data aggregation tree based on the source nodes selected by the proposed selection scheme. The objective of the method is to maximize the residual energy of the tree with minimized number of nodes from the source nodes to sink node. Simulation results show that proposed methods outperformance than other methods in terms of the number of active nodes involved for tracking, the accuracy of the tracking.

6.2. Future Research Works

In **Chapter 3**, we propose a localized approach for evolutionary algorithm, which can solve the optimization problem in a distributed way. We give two examples of suitable application of LEA and apply the approach to solve the coverage problem in WSNs. Nevertheless, it is still challenging to apply the proposed approach to solve

some other optimization problems in WSNs. Another potential issue is to consider how to design the coordination part more reasonable. In the work, the problem consists of subproblems. The coordination part is the most important part that the local information is exchanged for obtaining the local optimal solution of each subproblem in this part. However, as to WSNs, the energy efficiency is the fundamental requirement. How to use less communication cost to obtain the local optimal solution in the coordination part is a worthwhile future research direction.

In **Chapter 4**, we propose a coverage-preserving routing algorithm in order to maximize the sensing coverage of the routing path. However, our work is a preliminary investigation for coverage-preserving routing problem in wireless networks. We only consider a routing path from a source node to a sink node. However, in WSNs, there may be multiple source nodes and sink nodes. In multicast routing, the calculation of sensing coverage of the routing path will be more complex. The current method for selection of path nodes will not be effective for solving the multicast routing problem. Research into the coverage-preserving multicast routing problem is necessary and promising.

In **Chapter 5**, we design a localized selection scheme and dynamic construction of data aggregation tree for tracking the target in an energy-efficient way. For tracking continuously, when the aggregation tree is reconstructed, how to reconstruct the tree as soon as possible is our primary goals. However, as we all know, saving energy is the most important issue in WSNs. Thus, how to consider the energy saving of the nodes and reconstruct the data aggregation tree in real time will be one of the possible future directions on the topic of dynamic construction of data aggregation tree.

References

[AB11] A. Abdelgawad and M. Bayoumi, Low Power Distributed Kalman Filter for Wireless Sensor Networks, In *EURASIP Journal on Embedded Systems*, 2011.

[AK04] J. Al-karaki and A. Kamal, Routing techniques in wireless sensornetworks: A survey, In*IEEE WirelessCommunications*, vol. 11, no. 6, pp.6–28, December 2004.

[AK04] J. N. Alkaraki and A. E. Kamal, Routing techniques in wireless sensor networks: A survey, In *IEEE Transaction on Wireless Communications*, vol. 11, no. 6, pp. 6–28, December 2004.

[AL08] L. Arienzo and M. Longo, An energy-efficient strategy for target tracking through wireless sensor networks, In *Gruppo Telecomunicazioni Teoria dell'Informazione Conference*, Florence, Italy, June 2008.

[AM02] M. Arulampalam and S. Maskell et al, A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking, In *IEEE Trans on Signal Processing*, vol:50(2), pp:174-188, 2002.

[AMO93] R. Ahuja, T. Magnanti, and J. Orlin, Network Flows: Theory, Algorithms, and Applications, 1st ed. Prentice Hall, vol. 5, 1993.

[AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, Network Flows: Theory, Algorithms, and Applications, 1st ed. Prentice Hall, 1993, vol. 5.

[AS02] S. Akyildiz and S. Sankarasubramaniam et al, A survey on sensor networks, In *IEEE Communication Magazine*, pp: 102-114, 2002.

[ASS+02] I. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, Wireless Sensor Networks: A Survey. In *Elsevier Computer Networks*, pp. 393-422, 2002.

[AY02] K. Akkaya and M. Younis, A Survey of Routing Protocols in Wireless Sensor Networks, In *Elsevier Ad Hoc Network Journal*, pp.325-349, 2005

[BB03] H. Blom and E. Bloem, Tracking Multiple Maneuvering Targets by Joint Combinations of IMM and PDA, In *Proc. 42nd IEEE Conf. Decision and Control*, 2003.

[BC06] S. Babu and C. Kumar et al, Sensor Networks for Tracking a Moving Object using Kalman Filtering, In *IEEE International Conference on Industrial Technology*, pp:1077-1082, 2006.

[C04] M. Coates, Distributed Particle Filters for Sensor Networks, In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, 2004.

[CG00] E. Cant-Paz and D.E. Goldberg, Efficient parallel genetic algorithms: theory and practice, In *Computer Methods in Applied Mechanics and Engineering*, 186(9):221–238, June 2000.

[CH03] W.P. Chen and J.C. Hou et al, Dynamic Clustering for Acoustic Target Tracking in Wireless Sensor Networks, In *Proc. IEEE Int'l Conf. Network Protocols (ICNP)*, 2003.

[CI02] K. Charabarty and S. Iyengar et al, Grid coverage for surveillance and target location in distributed sensor networks, In *IEEE Transaction on Computers*, vol: 51, pp:1448-1453, 2002.

[CI05] M. Coates and G. Ing, Sensor network particle filters: motes as particles, In *Proceedings of IEEE Workshop on Statistical Signal Processing*, 2005.

[CJ02] B. Chen and K. Jamieson, Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks, *ACM Wireless Networks Journal*, Vol:8(5), pp: 481-494, 2002.

[CS11] J. Chen and M. Salim et al, A Single Mobile Target Tracking in Voronoi-based Clustered Wireless Sensor Network, In *Journal of Information Processing Systems*, Vol.7, No.1, March 2011.

[CT05] M. Cardei and M. Thai et al, Energy-efficient target coverage in wireless sensor networks, In *24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, 2005.

[D04] T. M. Dorigo, Ant colony optimization, MIT Press, USA, 2004.

[DB03] Dumitrescu and N. Boland, Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem, In *Networks*, vol. 42, no. 3, pp. 135–153, Oct 2003.

[DC03] S. Dhillon and K. Chakrabarty, Sensor placement for effective coverage and surveillance in distributed sensor networks, In *Wireless Communications and Networking (WCNC)*, pp: 1609-1614, 2003.

[DL07] A. Donka and M. Lyudmila, Extended Object Tracking Using Mixture Kalman Filtering, In *Numerical Methods and Applications*, pp:122–130, 2007.

[FF95] C. Fonseca and P. Fleming, An overview of evolutionary algorithms in multiobjective optimization. Evolutionary Algorithm, pp: 1–16, 1995.

[FS80] T. Fortmann and Y. Shalom et al, Multi-target tracking using joint probabilistic data association, In *19th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, pp: 807 – 812, Dec. 1980.

[FZ02] Q. Fang and F. Zhao et al, Counting Targets: Building and Managing Aggregates in Wireless Sensor Networks, In *Palo Alto Research Center (PARC) Technical Report P2002-10298*, June 2002.

[FZ03] Q. Fang and F. Zhao, Lightweight Sensing and Communication Protocols for Target Enumeration and Aggregation, In *ACM Symp on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003.

[GM04] C. Gui and P. Mohapatra, Power conservation and quality of surveillance in target tracking sensor networks, In *ACM the 10th Annual International Conference on Mobile Computing and Networking (Mobi- Com)*, pp:129–143, 2004.

[GMR08] J. Goncalves, J. Mendes, and M. Resende, A geneticalgorithm for the resource constrained multi-project schedulingproblem, In*European Journal of Operational Research*, pp: 1171–1190, 2008.

[GS93] N. Gordon and D. Salmond et al, Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation, In *IEE Proceedings F In Radar and Signal Processing*, IEE Proceedings F, Vol. 140, 1993.

[HB07] M. Hefeeda and M. Bagheri, Wireless Sensor Networks for Early Detection of Forest Fires, In *IEEE International Conference on Mobile Adhoc and Sensor Systems*, 8-11 Oct, 2007.

[HC02] C. Hue and J. Cadre, Sequential Monte Carlo Methods for Multiple Target Tracking and Data Fusion, In *IEEE Transaction on Signal Processing*, vol:50(2), 2002.

[HI04] K. Hwang and J. In et al, Dynamic sink oriented tree algorithm for efficient target tracking of multiple mobile sink users in wide sensor field, In *Vehicular Technology Conference*, Vol:7, pp:4607-4610, 2004.

[HK00] B. Hendrickson and T.G. Kolda, Graph partitioning models for parallel computing. In *Parallel Computing*, pages 1519-1534, 2000.

[HK00] J. Han and M. Kamber, Data Mining: Concepts and Techniques, The Morgan Kaufmann Series in Data Management Systems, Jim Gray (Ed.), Morgan Kaufmann Publishers, 2000.

[HK06] T. He and S. Krishamurthy et al, Vigilnet: An Integrated Sensor Network System for Energy-Efficient Surveillance, In *ACM Trans on Sensor Networks*, vol:2(1), pp:1-38, 2006.

[HK09] M. Hussain and P. Khan et al, WSN research activities for military application, In*Proceedings of the 11th international conference on Advanced Communication Technology*, 2009.

[HR10] A. Hess and A. Rantzer, Distributed Kalman Filter Algorithms for Self-localization of Mobile Devices, In *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, 2010.

[HS04] T. He and S. Krishnamurthy, et al, Energy-Efficient Surveillance System Using Wireless Sensor Networks, In *MobiSys'04*, 2004.

[HS09] M. Hung and W. Sheng, Multi-target Tracking and Observing in Mobile Sensor Networks, In *Conference on Theoretical and Applied Computer Science (TACS09)*, October 24th, 2009.

[HT04] C. Huang and Y. Tseng et al, The Coverage Problem in Three-Dimensional Wireless Sensor Networks, In *IEEE Global Telecommunications Conference*, Vol: 5, pp: 3182-3186, 2004.

[HV07] T. He and P. Vicaire et al, Achieving Real-Time Target Tracking Using Wireless Sensor Networks, In *ACM Transaction on Embedded Computing System (TECS)*, pp: 37-48, 2007.

[HZD07] K. He, L. Zheng, S. Dong, J. L. Tang, and C. Zheng, Pgo: A parallel computing platform for global optimization based on genetic algorithm, In *Computers and Geosciences*, 33(3):357–366, March 2007.

[I08] S. Ismaell, Adaptive combination of forecasts with application to wind energy, In *International Journal of Forecasting*, page 679-693, 2008.

[IC05] G. Ing and M. Coates, Parallel particle filters for tracking in wireless sensor networks, In *IEEE 6th Workshop on Signal Processing Advances in Wireless Communications*, 5-8 June 2005.

[IY04] H. Ishibuchi and T. Yamamoto, Fuzzy rule selection by multiobjectivegenetic local search algorithms and rule evaluationmeasures in data mining, In *Fuzzy Sets and Systems*, pp: 59–88,January 2004.

[JK11] D. Jeswani and A. Kesharwani, Efficient Target Tracking through Binary-Detection in Sparsely Deployed WSN, In *IEEE Projects*, 2011.

[JRM00] A. Jain, R. Duin, and J. Mao, Statistical pattern recognition:A review, In *IEEE Transactions on Pattern Analysis andMachine Intelligence*, pp: 4–37,January 2000.

[JS10] A. Javier and G. Sanchez, Wireless Sensor Network Deployment for Monitoring Wildlife Passages, In *Sensors*, 10(8), pp:7236-7262, 2010.

[JW04] D. Jourdan and O. Weck, Layout optimization for a wirelesssensor network using a multi-objective genetic algorithm, In *Proceedingsof the IEEE 59th Vehicular Technology Conference*,pp: 2466–2470, Jan 2004.

[KA08] M. Kushwaha and I. Amundson, Multi-modal Target Tracking using Heterogeneous Sensor Networks, In *17th International Conference on Computer Communications and Networks (ICCCN 08)*, 2008.

[KKM00] K. Kojima, W. Kawamata, H. Matsuo, and M. Ishigame, Network based parallel genetic algorithm using client-server model, pp: 244-250, 2000.

[KL04] S. Kumar and T. Lai et al, On k-coverage in a mostly sleeping sensor network, In *ACM International Conference on Mobile Computing and Networking (MobiCom)*, pp:114–158, 2004.

[KS08] S. Kim and J. Seo, Wireless Sensor Network based Asset Tracking Service, In *PICMET 2008 Proceedings*, pp: 27-31, July 2008.

[KV03] H. Kung and D. Vlah, Efficient location tracking using sensor networks, In *Wireless Communications and Networking (WCNC)*, 2003.

[KW08] M. H. Kalos and P. A. Whitlock, Monte Carlo Methods, second revised and enlarged edition ed. Weinheim, Bergstr, 2008, vol. 2, no. 9.

[KW08] M. Kalos and P. Whitlock, Monte Carlo Methods, second revised and enlarged editioned. Weinheim, Bergstr, vol. 2, no. 9, 2008.

[LC04] J. Liu and M. Chu et al, Distributed State Representation for Tracking Problems in Sensor Networks, In *Information Processing in Sensor Networks*, pp:234- 242, 2004.

[LC07] J. Liu and M. Chu et al, Resource-Aware Multi-Target Tracking in Distributed Sensor Networks, In *IEEE Signal Processing Magazine Special Issue on Resource-Constrained Signal Processing*, Communications, and Networking, 2007.

[LC11] X. Liu and J. Cao et al, Energy efficient clustering for WSN-based structural health monitoring, In *IEEE INFOCOM*, pp: 2768 – 2776, 10-15 April 2011.

[LCZ10] Y. Liang, J. Cao, L. Zhang, R. Wang, Q. Pan, A Biologically Inspired Sensor Wakeup Control Method for Wireless Sensor Networks, In *IEEE Transactions on Systems, Man, and Cybernetics – Part C (TSMC)*, 40(5), pp:525-530, 2010.

[LD11] H. Lee and D. Kim, Mobile Embedded Health-Care System Working on Wireless Sensor Network, In *Third International Conference on Communications and Mobile Computing (CMC)*, pp:161-164, 18-20 April 2011.

[LK04] J. Lee and B. Krishnamachari, Impact of Heterogeneous Deployment on Lifetime Sensing Coverage in Sensor Networks, In *IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, 2004.

[LP06] C. Lin and W. Peng et al, Efficient In-Network Moving Object Tracking in Wireless Sensor Networks, In *IEEE Transaction on Mobile Computing*, Vol:5(8), pp:1044-1056,2006.

[LTK07] C.C. Lai, C.K. Ting, and R.S. Ko, An effective genetic algorithm to improve wireless sensor network lifetime for large-scale surveillance applications, In *Proceedings of IEEE Congress on Evolutionary Computation (CEC'07)*, pages 3531–3538, 2007.

[M10] R. Mulligan, Coverage in Wireless Sensor Networks: A Survey, In *Network Protocols and Algorithms*, Vol. 2, No. 2, 2010.

[MA08] P. Mahdy and O. Ali et al, Enhanced object tracking with received signal strength using Kalman filter in sensor networks, In *International Symposium on Telecommunications*, pp:318-323, 2008.

[MI02] S. Mcginnity and G. Irwin, Multiple Model Bootstrap Filter for Maneuvering Target Tracking, In *IEEE Transaction on Aerospace and Electronic Systems*, vol:36(3), 2002.

[MKC09] H. Min, H. Ko, and S. Chang, A genetic algorithm approachto developing the multi-echelon reverse logistics network forproduct returns. In *omega*, pages 59–69, 2006.

[MKP+05] S. Megerian, F. Koushanfar, M. Potkonjak, and M. Srivastava, Worst andbest-case coverage in sensor networks, In*IEEE Transactions on MobileComputing*, vol. 4, no. 1, pp. 84–92, Jan-Feb 2005.

[MKP05] S. Megerian, F. Koushanfar, M. Potkonjak, and M. Srivastava, Worst and best-case coverage in sensor networks, In *IEEE Transactions on Mobile Computing*, vol. 4, no. 1, pp. 84–92, Jan-Feb 2005.

[MM05] V. Manfredi and S. Mahadevan et al, Switching Kalman Filters for Prediction and Tracking in an Adaptive Meteorological Sensing Network, In *IEEE Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, 2005.

[MP08] H. Medeiros and J. Park et al, Distributed Object Tracking Using a Cluster-Based Kalman Filter in Wireless Camera Networks, In *IEEE Journal of Selected Topics in Signal Processing*. 4, 2, 448-463, 2008.

[MS10] M. Mansouri and H. Snoussi, A Sensor Selection Method for Target Tracking in Wireless Sensor Networks using Quantized Variational Filtering, In *IEEE 72nd Vehicular Technology Conference Fall*, 2010.

[MS+10] M. Kouakou and S. Yamamoto et al, "Deployment Planning Tool for Indoor 3D-WSNs", *Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing*, 2010.

[NF08] T. Naumowicz and R. Freeman et al, Autonomous monitoring of vulnerable habitats using a wireless sensor network, In *Proceedings of the workshop on Real-world wireless sensor networks*, 2008.

[NS08] A. Norris and M. Saafi, Temperature and moisture monitoring in concrete structures using embedded nanotechnology/microelectromechanical systems (MEMS) sensors, Constr. Build. Mater. 22, 111-120, 2008.

[OB08] L. Ong and T. Bailey, Decentralised Particle Filtering for Multiple Target Tracking in Wireless Sensor Networks, In *11th International Conference on Information Fusion*, 2008.

[OLP03] P. Osmera, B. Lacko, and M. Petr, Parallel evolutionary algorithms, In *Proceeding of Computational Intelligence in Robotics and Automation*, pages 1348-1353, July 2003.

[R10] K. Rabindra, Multi-Objective Node Placement Methodology for Wireless Sensor Network, In *IJCA Special Issue on "Mobile Ad-hoc Networks" MANETs*, 2010.

[RE07] H. Rowaihy and S. Eswaran et al, A survey of sensor selection schemes in wireless sensor networks, In *SPIE Defense and Security Symposium Conference on Unattended Ground*, 2007.

[RE07] H. Rowaihy and S. Eswaran et al, A survey of sensor selection schemes in wireless sensor networks, In *SPIE Defense and Security Symposium Conference on Unattended Ground*, 2007.

[S04] J.A. Stankovic, Research Challenges in Wireless Sensor Networks, In *Special issue on embedded sensor networks and wireless computing*, Volume 1, Issue 2, July 2004.

[S05] R. Olfati-Saber, Distributed Kalman Filter with Embedded Consensus Filters, In *44th IEEE Conference onDecision and Control*, 2005.

[SK11] R. Sangeetha and B. Kalpana, Denoising the signals using Kalman filter for target tracking in wireless sensor networks, In *3rd International Conference on Electronics Computer Technology (ICECT)*, 2011.

[SL93] Y. Shalom and X. Li, Estimation and Tracking: Principles Techniques and Software, In *Artech House*, 1993.

[SP01] S. Slijepcevic and M. Potkonjak, Power efficient organization of wireless sensor networks, In *Proceedings of IEEE International Conference on Communications(ICC'01)*, 2:472–476, 2001.

[ST05] X. Shan and J. Tan, Mobile sensor deployment for a dynamic cluster-based target tracking sensor network, In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp:1452-1457, 2005.

[TB09] Y. Tai and Y. Bo, Collaborative Target Tracking in Wireless Sensor Network, In *9th International Conference on Electronic Measurement & Instruments*, 16-19 Aug, 2009.

[TL09] V. Tseng and E. Lu, Energy-efficient real-time object tracking in multi-level sensor networks by mining and predicting movement patterns, In *Journal of Systems and Software*, vol:82(4), pp:697-706, 2009.

[TR09] M. Tinati and T. Rezaii, Multi-target Tracking in Wireless Sensor Networks Using Distributed Joint Probabilistic Data Association and Average Consensus Filter, In *ICACC '09 Proceedings of the International Conference on Advanced Computer Control*, 2009.

[TS08] J. Teng and H. Snoussi et al, Variational Filtering Algorithm For Interdependent Target Tracking and Sensor Localization in Wireless Sensor Network, In *16th European Signal Processing Conference*, pp: 1-5, 2008.

[TS11] J. Teng and H. Snoussi, Collaborative multi-target tracking in wireless sensor networks, In *International Journal of Systems Science*, Volume 42, Issue 9, pp: 1427-1443, 2011.

[VN01] G. Veres and J. Norton, Improved Particle Filter for Multitarget- Multisensor Tracking with Unresolved Applications, In *IEE Target Tracking: Algorithms and Applications*, vol:1, pp:12/1-12/5, 2001.

[WB01] G. Welch and G. Bishop, An Introduction to the Kalman Filter, In *SIGGRAPH 2001 Course*.

[WB08] Z. Wang and E. Bulut et al, A Distributed Cooperative Target Tracking with Binary Sensor Networks, In *IEEE International Conference on Communications Workshops*, pp:306 – 310, 2008.

[WC07] C. Wu and Y. Chung, Heterogeneous Wireless Sensor Network Deployment and Topology Control Based on Irregular Sensor Model, In *GPC'07 Proceedings of the 2nd international conference on Advances in grid and pervasive computing*, 2007.

[WF07] B. Wang and C. Fu, Layered Diffusion based Coverage Control in Wireless Sensor Networks, In *32nd IEEE Conference on Local Computer Networks*, pp:504-511, Oct 2007.

[WH08] Y. Wang and C.Hu et al, Efficient Placement and Dispatch of Sensors in a Wireless Sensor Network, In *IEEE Transaction on Mobile Computing*, Vol:7(2), pp: 262-274, 2008.

[WJ05] G. Werner-Allen and J. Johnson et al, Monitoring volcanic eruptions with a wireless sensor network, In *Wireless Sensor Networks, Proceedings of the Second European Workshop*, pp. 108–120, Jan, 2005.

[WR10] J. Wang and X. Ren et al, A Remote Wireless Sensor Networks for Water Quality Monitoring, In *cicc-itoe*, pp: 7-12, 2010.

[WW06] Y. Wand and D. Wang et al, Hops-based Sleep Scheduling Algorithm for Enhancing Lifetime of Wireless Sensor Networks, In *IEEE International Conference on Mobile Adhoc and Sensor Systems*, pp:709-714, 2006.

[WX03] X. Wang and G. Xing et al, Integrated coverage and connectivity configuration in wireless sensor networks, In *Conference on Embedded Networked Sensor Systems*. pp:28–39, 2003.

[WX06] L. Wang and Y. Xiao, A survey of energy-efficient scheduling mechanisms in sensor networks, In *Mobile Networks and Applications*, vol:11(5), pp:723-740, 2006.

[WZ09] J. Wei and X. Zhang, Mobile Multi-Target Tracking in Two-Tier Hierarchical Wireless Sensor Networks, In *IEEE Military Communications Conference*, pp: 1-6, 18-21 Oct. 2009.

[YF06] L. Yang and C. Feng et al, Adaptive Tracking in Distributed Wireless Sensor Networks, In *Proceedings of the 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems*, pp:103-111, March 27-30, 2006.

[YK07] O. Younis and M. Krunz et al, Coverage Without Location Information, In *IEEE International Conference on Network Protocols ICNP*, pp:51-60, 2007.

[YT03] H. Yao and L. Tian. A genetic-algorithm-based selective principalcomponent analysis (ga-spca) method for high-dimensional datafeature extraction, In *IEEE Transactions on Geoscience andRemote Sensing*, pp: 1469– 1478, June 2003.

[YZ03] F. Ye and G. Zhong et al, PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks, In *Proceedings of the 23rd IEEE International Conference on Distributed Computing Systems (ICDCS 2003)*, pp:28-37, May 2003.

[Z10] X. Zhang, Decentralized Sensor-Coordination Optimization for Mobile Multi-Target Tracking in Wireless Sensor Networks, In I*EEE Global Telecommunications Conference*, pp:1-5, 6-10 Dec. 2010.

[Z96] Z. Michalewicz, Genetic algorithms + Data Structure = Evolution Programs, Springer-Verlag, Berline, 1996.

[ZC04] W. Zhang and G. Cao, DCTC: Dynamic Convoy Tree-Based Collaboration for Target Tracking in Sensor Networks, In *IEEE Transaction on Wireless Communication*, pp:1689-1701, 2004.

[ZC+04] W. Zhang and G. Cao, Optimizing Tree Reconfiguration for Mobile Target Tracking in Sensor Networks, In *INFOCOM*, 2004.

[ZC07] M. Zhang and M. Chan et al, Coverage Protocol for Wireless Sensor Networks Using Distance Estimates, In *4th Annual IEEE Communications Society Conference on Sensor*, Mesh and Ad Hoc Communications and Networks, 2007.

[ZC+07] Y. Zou and K. Chakrabarty, Sensor deployment and target localization in distributed sensor networks, In *ACM Trans on Embedded Computing Systems*, Vol:1(3), pp: 61-91, 2007.

[ZH05] H. Zhang and J. Hou, Maintaining sensing coverage and connectivity in large sensor networks, In *Wireless ad hoc and Sensor Networks,* Vol:1, pp:89–123, 2005.

[ZS08] X. Zhu, and R. Sarkar, et al, Light-Weight Contour Tracking in Wireless Sensor Networks, In *27th IEEE Conference on Computer Communications (INFOCOM 2008)*, 2008.

[ZY06] J.Zhang and T.Yan et al, Deployment Strategies for Differentiated Detection in Wireless Sensor Networks, In *3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, pp:316-325, 2006.

[ZZ09] Z. Zhong and T. Zhu et al, Tracking with Unreliable Node Sequences, In *INFOCOM*, 2009.