CHAPTER 2

RELATED LITERATURE SURVEY

2.1 INTRODUCTION

WSN are emerging as essential and popular ways of providing pervasive computing environments for various applications. In all these environments energy constraint is the most critical problem that must be considered. Since the radio transmission and reception consumes lot of energy, one of the important issues in wireless sensor network is the inherent limited battery power within network sensor nodes. Therefore, battery power is crucial parameter in the algorithm design, to increase lifespan of nodes in the network.

To overcome the above problem, clustering is introduced to WSN because of its network scalability, energy-saving attributes and network topology stabilities. A cluster head can be one of the nodes or specifically richer in resources. The general number of cluster heads within the network and the amount of nodes per cluster can be variable or fixed by the user. Cluster heads can also form a second tier network, i.e. creating another level of hierarchy or they can just pass on the data to the base station (Jennifer Yick et al 2008). However, there also exist some disadvantages associated with individual clustering scheme, such as additional overheads during cluster head (CH) selection, assignment and cluster construction process. In addition to maximizing the lifespan of sensor nodes, it is preferable to distribute the
energy dissipated throughout the wireless sensor network in order to maximize overall network performance.

Much research has been done in recent years, investigating different aspects like, low power protocols, network establishments, routing protocol, and coverage problems of wireless sensor networks. This chapter discusses and compares several aspects and characteristics of some widely explored clustering algorithms in WSN, e.g. attributes, metrics, advantages and disadvantages of corresponding clustering algorithms and presents a meta–heuristics optimization algorithm for CH selection for WSN.

2.2 CLUSTERING ALGORITHMS FOR WSN

Energy balancing and network lifetime of WSN have drawn much attention in recent years. Many previous energy efficient mechanisms adopt different means to balance energy consumption among sensor nodes to prolong network lifetime. There are number of clustering protocols have been proposed in literature e.g. LEACH, HEED, DWEHC, PANEL, UCS, EECS, EEUC, PEGASIS, and TEEN. The cluster formation overhead of the clustering protocols includes packet transmission cost of the advertisement, node joining and leaving, and scheduling messages from sensor nodes. This section describes some of the existing popular clustering algorithms and energy efficient clustering protocols to select the efficient cluster heads in WSN (Wang et al 2010).

2.2.1 Low Energy Adaptive Clustering Hierarchy (LEACH)

LEACH algorithm divides wireless sensor network into several clusters. The algorithm introduces a random clustering scheme for wireless sensor network (Heinzelman et al 2000). It is a dynamic clustering routing method where nodes are selected as cluster head randomly based on a
threshold value calculated using Equation 2.1. Main techniques of LEACH protocol include algorithms for distributing cluster forming, adaptive cluster forming, and cluster header position changing. The technique of distributing cluster forming ensures self-organization of most target nodes. The adaptive cluster forming and cluster header position changing algorithms ensure to share the energy dissipation fairly among all nodes and prolong the lifetime of the whole system in the end.

In LEACH, the nodes organize themselves into local clusters, with one node acting as the CH. All non cluster head nodes must transmit their data to the cluster head, while the cluster head node must receive data from all the cluster members, perform signal processing functions on the data e.g. data aggregation and transmit data to the remote base station. Therefore being a cluster head node is much more energy intensive than being a non cluster head node. In the scenario where all nodes are energy limited, if the cluster heads were chosen a priori and fixed throughout the system lifetime, the cluster head sensor nodes would quickly use up their limited energy, such method is known as static clustering method. Once the cluster head runs out of energy, it is no longer operational. Thus, when a cluster head node dies (e.g. uses up all its battery energy) all the nodes that belong to the cluster lose communication ability.

Thus, LEACH incorporates randomized rotation of the high energy cluster head position such that it rotates among the sensors in order to avoid draining the battery of any one sensor in the network. In this way the energy load associated with being a cluster head is evenly distributed among the nodes. Media access in LEACH was chosen to reduce energy dissipation in the non cluster head nodes. Since the cluster head node knows all the cluster members, it can create a TDMA schedule that tells each node exactly when to transmit its data. This allows the nodes to remain in the sleep state with
internal modules powered down, as long as possible. In addition using a TDMA schedule for data transfer prevents intra cluster collisions.

The operation of LEACH is divided into rounds. Each round begins with a set up phase when the clusters are organized followed by a steady state phase where several frames of data are transferred from the nodes to the cluster head and on to the base station. The nodes must all be time synchronized in order to start the set up phase at the same time. In order to minimize the set up overhead, the steady state phase is long compared to the set up phase.

2.2.1.1 Determining cluster head nodes

Initially, when clusters are being created, each node decides whether or not to become a cluster head for the current round. This decision is based on the suggested percentage ‘p’ of cluster heads for the network (determined a priori) and the number of times the node has been a cluster head so far. This decision is made by the node n choosing a random number between 0 and 1. If the number is less than a threshold T(n), the node becomes a cluster head for the current round (Haiming Yang & Biplab Sikdar 2007). The threshold is set as in Equation 2.1.

\[
T(n) = \begin{cases} 
  P & \text{if } n \in G \\
  1 - P \left( r \mod \left( \frac{1}{p} \right) \right) & \text{otherwise}
\end{cases}
\]

(2.1)

where, \( p \) is desired percentage of cluster heads, \( r \) is the current round, \( G \) is the set of nodes that have not been cluster heads in the last \( \left( \frac{1}{p} \right) \) rounds.
Using this threshold, each node will be a cluster head at some point within \( \left( \frac{1}{p} \right) \) rounds. During round 0 (\( r = 0 \)), each node has a probability ‘\( p \)’ of becoming a cluster head. The nodes that are cluster heads in round 0 cannot be cluster heads for the next \( \left( \frac{1}{p} \right) \) rounds. Thus the probability that the remaining nodes are cluster heads must be increased, since there are fewer nodes that are eligible to become CHs. After \( \left( \frac{1}{p} \right) \) -1 rounds, \( T = 1 \) for any nodes that have not yet been clusterheads, and after \( \left( \frac{1}{p} \right) \) rounds, all nodes are once again eligible to become clusterheads. Future versions of this work will include an energy-based threshold to account for non-uniform energy nodes. In this case, it is assumed that all nodes begin with the same amount of energy and being a cluster head removes approximately the same amount of energy for each node. However LEACH was one of the best approaches for cluster head selection and energy balancing, but still it has certain limitations as describe in the next section. LEACH is a completely distributed approach and requires no global information of network. There are many variants and modification of LEACH developed, which form LEACH family. Some of the modified LEACH (Fan & Song 2007) discussed in the various existing works are Two-Levels Hierarchy for Low-Energy Adaptive (TL-LEACH) (Loscri et al 2005), Energy-LEACH (E-LEACH) (Fan & Song 2007), Centralized Low Energy Adaptive Clustering Hierarchy (CLEACH) (Heinzelman et al 2002), Vice-LEACH (V-LEACH) (Yassein et al 2009), Low Energy Adaptive Clustering Hierarchy Fuzzy Logic (LEACH-FL) (Ran et al 2010), Weighted Low Energy Aggregation Clustering Hierarchy (W-LEACH), (Abdulsalam & Kamel 2010) and Threshold-based LEACH protocol (T-LEACH) (Hong et al 2009).
2.2.1.2 Limitation of LEACH algorithm

Though, the energy consumption is distributed among all the sensor nodes, so many limitations of LEACH are found as follows:

- There is a possibility of none of the sensor node in the network selecting itself as a cluster head during some rounds.
- There is a possibility of concentration of all the selected cluster heads in only a part of the network.
- The even distribution of the cluster heads is not guaranteed.
- The clusters are not guaranteed to be of equal sizes.
- A balanced cluster head distribution is not guaranteed.
- Cluster head selection is not energy adaptive.
- Cluster formation, during each round, consumes energy of all the sensor nodes.
- Nodes near the cluster boundary are expected to consume more energy as compared to other sensor nodes.

Most of these limitations seem to be encouraged by the basic cluster head selection strategy, of generation of a random number and its comparison with a calculated probabilistic threshold by all the sensor nodes, used in LEACH. The possibility of the random numbers to be generated being either greater or smaller to their respective calculated thresholds may cause
none or all of the eligible sensor nodes to select themselves as cluster heads, during some data gathering rounds. In such scenario, all nodes acting as forced cluster heads are required to transmit their sensed and processed data, directly to the distant base station and result in their more energy consumption which, as its consequence, reduces the network lifetime (Younis et al 2003).

2.2.2 Hybrid Energy Efficient Distributed (HEED) Clustering

Hybrid Energy Efficient Distributed clustering (HEED), introduced by Younis & Fahmy (2004), is a multi-hop WSN clustering algorithm which brings an energy-efficient clustering routing with explicit consideration of energy. Different from LEACH, in the manner of CH election, HEED does not select nodes as CHs randomly. The manner of cluster construction is performed based on the hybrid combination of two parameters. One parameter depends on the node’s residual energy, and the other parameter is the intra-cluster communication cost. In HEED, elected CHs have relatively high average residual energy compared to member nodes. Additionally, one of the main goals of HEED is to get even-distributed CHs throughout the networks. Moreover, despite the phenomena that two nodes, within each other’s communication range, become CHs together, but the probability of this phenomena is very small in HEED.

The advantages of the HEED protocol are as follows: (1) It is a fully distributed clustering method that benefits from the use of the two important parameters for CH election; (2) Low power levels of clusters promote an increase in spatial reuse while high power levels of clusters are required for inter-cluster communication. This provides uniform CH distribution across the network and load balancing; (3) Communications in a multi-hop fashion between CHs and the BS promote more energy conservation and scalability in contrast with the single-hop fashion, i.e., long-
range communications directly from CHs to the sink, in the LEACH protocol (Taheri et al 2012).

However, there are some limitations with HEED as follows: (1) The use of tentative CHs that do not become final CHs leave some uncovered nodes. As per HEED implementation, these nodes are forced to become a CH and these forced CHs may be in range of other CHs or may not have any member associated with them. As a result, more CHs are generated than the expected number and this also accounts for unbalanced energy consumption in the network (Aslam et al 2011). (2) Similar to LEACH, the performing of clustering in each round imposes significant overhead in the network. This overhead causes noticeable energy dissipation which results in decreasing the network lifetime. (3) HEED suffers from a consequent overhead since it needs several iterations to form clusters. Each iteration, lot of packets is broadcast. (4) Some CHs, especially near the sink, may die earlier because these CHs have more work load, and the hot spot will come into being in the network (Wei et al 2011).

2.2.3 Distributed Weight based Energy Efficient Hierarchical Clustering (DWEHC) Algorithm

Distributed Weight based Energy Efficient Hierarchical Clustering (DWEHC)algorithm, proposed by Ding et al (2012) is a distributed clustering algorithm similar to HEED. The main objective of DWEHC is to improve HEED by building balanced cluster sizes and optimize the intra-cluster topology using location awareness of the nodes. Both DWEHC and HEED share some similarities including no assumptions about network size and density, and considering residual energy in the process of CH election. Every node implements DWEHC individually and the algorithm ends after several iterations that are implemented in a distributed manner. Different from
LEACH and HEED, DWEHC creates a multi-level structure for intra-cluster communication and limits a parent node’s number of children. Moreover, the only locally calculated parameter weight is defined for CH election in DWEHC.

The following is the advantages of DWEHC: (1) Like HEED, it is a fully distributed clustering method that is based on a function of the sensor’s energy reserve and the proximity to the neighbors for CH election; (2) Considering energy reserves in CH election, DWEHC generates more well-balanced CHs distribution and achieves significantly lower energy consumption in intra-cluster and inter-cluster routing than HEED; (3) The clustering process of DWEHC terminates in a few iterations, and does not depend on network topology or size.

Some disadvantages of DWEHC are summarized as follows: (1) Similar to LEACH, single-hop inter-communication, directly from CHs to the BS, is performed in DWEHC. Thus DWEHC may result in significant amount of energy consumption, and is not applicable to large-region networks; (2) In the process of cluster formation; the iterative nature in both DWEHC and HEED produces a relatively high control message overhead compared to other protocols.

2.2.4 Position based Aggregator Node Election (PANEL) Algorithm

Position based Aggregator Node Election (PANEL) algorithm, presented by Buttyan & Schaffer (2007), is a position-based clustering routing protocol for WSN. With respect to other CH election protocols, PANEL supports asynchronous sensor network applications where the sensor node readings are fetched by the BSs. The main goal of PANEL is to elect aggregators, i.e., CHs, for reliable and persistent data storage applications Buttyan & Schaffer (2010).
The following are the main merits of PANEL: (1) This protocol is an energy-efficient protocol that ensures load balancing because each node is elected aggregator, i.e., CH, nearly equally frequently. Besides, data aggregation is performed and communication load is reduced, accordingly PANEL can prolong the network lifetime; (2) The outstanding feature of PANEL that makes it different from other data-aggregation based clustering protocols is that besides synchronous scenes, it also supports asynchronous applications.

The main limitations of PANEL are discussed as follows: (1) The assumption that the clusters are determined before deployment and thus cannot be applied to WSN dynamics; (2) Geographical position information of the nodes is used to determine which node should be the aggregators. This is a restriction in WSN, because the geographical position is not always available without special condition, such as GPS-like hardware and software; (3) A crucial assumption of PANEL, described by the authors of PANEL, is that the nodes within a cluster form a connected sub-network. If this assumption is not satisfied, and the sub-network within a cluster is partitioned, then some nodes will not hear the announcement of the node closest to the reference point, and they will elect another node as aggregator.

2.2.5 Unequal Clustering Size (UCS)

Unequal Clustering Size (UCS) model was proposed by Soro & Heinzelman 2005 for network organization in order to balance energy consumption of CHs, thus increasing the network lifetime. UCS is the first unequal clustering model for WSN organization. It is assumed that the positions of the CHs are determined a priori, with all CHs arranged symmetrically in concentric circles around the BS which is located in the
center of the network, thus it is easy to control the actual sizes of different clusters.

The advantages of UCS are discussed as follows: (1) By changing the number of nodes in every cluster with respect to the expected communication load, UCS can maintain more uniform energy consumption among the CHs. Therefore, the total energy dissipated for every CH is similar and UCS can prolong network lifetime compared with the model of Equal Clustering Size (ECS); (2) Using the two-layered network model and two-hop inter-cluster communication method, UCS results in a shorter average transmission distance compared with LEACH, thus effectively reduces the total energy consumption.

Limitations in UCS are as follows: (1) UCS is constrained by the assumption that the network is heterogeneous, and CHs are performed by super nodes all the time and are deployed at pre-determined locations. That is to say, it lacks universality (Ren et al 2010); (2) CHs are required to locate in the center of the cluster, thus a key factor, residual energy of nodes, is not considered in UCS; (3) Similar to TL-LEACH, despite that the average transmission distance is decreased in comparison with LEACH, the two-hop inter-cluster routing of UCS is still not applicable to large-range networks, because it uses only two hops for data transmission from sources to the BS, and long-distance communications need much energy consumption.

2.2.6 Energy Efficient Clustering Scheme (EECS)

Energy Efficient Clustering Scheme (EECS), proposed by Ye et al (2006), is a clustering algorithm which better suits the periodical data gathering applications. EECS is a LEACH-like scheme, where the network is partitioned into several clusters and single-hop communication between the CH and the BS is performed. In EECS, CH candidates compete for the ability
to elevate to CH for a given round. This competition involves candidates broadcasting their residual energy to neighboring candidates. If a given node does not find a node with more residual energy, it becomes a CH. Different from LEACH for cluster formation; EECS extends LEACH by dynamic sizing of clusters based on cluster distance from the BS.

The advantages of EECS are summarized as follows: (1) Based on energy and distance, EECS constructs balancing point between intra-cluster energy consumption and inter-cluster communication load; (2) Clustering is performed by dynamic sizing based on cluster distance from the BS. This addresses the problem that clusters with a larger distance to the BS require more energy for transmission than those with a shorter distance, and bring about low message overheads and uniform distribution of CHs compared to LEACH. However, there exist a few disadvantages in EECS as follows: (1) Account of single-hop communications in EECS, long-range transmissions directly from CHs to the BS can lead to much energy consumption. Hence it is not suitable for large-range networks; (2) EECS requires more global knowledge about the distances between the CHs and the BS, and the task of global data aggregation adds overheads to all sensor nodes; (3) EECS produces much more control overhead complexity because all nodes must compete for becoming CHs.

2.2.7 Energy Efficient Uneven Clustering (EEUC)

Energy Efficient Uneven Clustering (EEUC) algorithm, proposed by Li et al (2005), is a clustering and distributed competitive algorithm, where CHs are elected by localized competition, which is unlike LEACH. Every node has a pre-assigned competitive range, which is smaller as it gets close to the BS. This makes EEUC an unequal clustering approach for the purpose of balancing energy consumption among CHs and solving the hot spots problem.
During the process of CH election in EEUC, each node generates a random number, and only the node whose number is greater than a threshold will be activated for CH election by broadcasting compete message within a competition radius which is determined by its distance to the BS. According to above discussion, the advantages of EEUC are as follows: (1) To address the hot spots problem, EEUC introduces an unequal clustering mechanism to balance the energy consumption among CHs. Accordingly, the unequal clustering mechanism in EEUC improves the network lifetime over LEACH and HEED; (2) Based on communication cost, this protocol can save more energy via inter-cluster multi-hop routing mechanism in steady state phase, because a CH would choose its relay node from the two whose communication cost are the least among all of its neighbor CHs.

However, there are several drawbacks in EEUC as follows: (1) Performing of clustering in each round imposes significant overhead, because each node must broadcast and receive a large amount of competition message for CH election, even though most of them cannot win and most of the elected nodes are not suitable to be as CHs; (2) The extra global data aggregation can result in much overhead for all nodes and deteriorate the network performance; (3) The routing scheme can result in new hot spots, in that only one of the two nodes whose communication costs are the least among the neighbor CHs can be relay nodes, even though both of them have little residual energy.

2.2.8 Power Efficient Gathering in Sensor Information Systems (PEGASIS)

Power Efficient Gathering in Sensor Information Systems (PEGASIS), proposed by Lindsey et al (2002), is an improvement of LEACH. The main idea of PEGASIS is for each node to only communicate with their
close neighbors and take turns being the leader for transmission to the sink. In PEGASIS, the locations of nodes are random, and each sensor node has the ability of data detection, wireless communication, data fusion and positioning. Energy load is distributed evenly among the sensor nodes in the network (Jung et al 2007).

In PEGASIS, the nodes are organized to form a chain, which can either be concentratedly assigned by the sink and broadcast to all nodes or accomplished by the nodes themselves using a greedy algorithm. If the chain is formed by the nodes themselves, they can first get the location data of all nodes and locally compute the chain using the same greedy algorithm. During the process of chain formation in PEGASIS, it is assumed that all nodes have global knowledge of the network and the greedy algorithm is employed. The chain construction is commenced from the furthest node from the sink and the closest neighbor to this node will be the next node on the chain. When a node on the chain dies, the chain will be reconstructed in the same manner to bypass the dead node. For gathering data from sensor nodes in each round, each node receives data from one neighbor, fuses the data with its own, and transmits to the other neighbor on the chain. By moving from node to node, the fused data eventually are sent to the sink by the leader at a random position on the chain. The leader is important for nodes to die at random locations, in respect that the idea of nodes dying at random places is to enhance the robustness of the network (Lindesy & Raghavendra 2002).

The advantages of PEGASIS are as follows: (1) This protocol is able to outperform LEACH for different network sizes and topologies, because it reduces the overhead of dynamic cluster formation in LEACH, and decreases the number of data transmission volume through the chain of data aggregation; (2) The energy load is dispersed uniformly in the network. To ensure that the fixed sensor node is not select as the leader and thus to prevent
the subsequent early death of this sensor node, all sensor nodes act as the leader in turn (Chen & Lin 2012).

However, there are some disadvantages in PEGASIS: (1) It is the necessity of having a complete view of the network topology at each node for chain construction and that all nodes must be able to transmit directly to the sink. Thus, this scheme is unsuitable for those networks with a time varying topology (Fasolo et al 2007); (2) It is assumed that each sensor node can be able to communicate with the sink directly, but nodes usually use multi-hop communications with the sink in practical cases. Furthermore, long-range communications directly from the node to the sink can breed too much energy consumption; (3) The communication manner suffers from excessive delays caused by the single chain for distant nodes and a high probability for any node to become a bottleneck; (4) It is a difficult task for all nodes to maintain a complete database about the location of all other nodes in the network, furthermore the network is not very scalable because all nodes must have global knowledge of the network and employ the greedy algorithm.

2.2.9 Threshold Sensitive Energy Efficient Sensor Network (TEEN) Algorithm

Threshold Sensitive Energy Efficient Sensor Network (TEEN) algorithm, proposed by Manjeshwar & Agrawal (2001), is a hierarchical protocol whose main goal is to cope with sudden changes in the sensed attributes such as temperature. The protocol combines the hierarchical technique in line with a data-centric approach. The nodes sense their environment continuously, but the energy consumption in this algorithm can potentially be much less than that in the proactive network, because data transmission is done less frequently.
In TEEN, a CH sends its members a hard threshold and a soft threshold. Thus the hard threshold tries to reduce data communications by allowing the nodes to transmit only when the sensed attribute is in the range of interest. The soft threshold further reduces data communications might have otherwise occurred when there is little or no change in the sensed attribute. At the expense of increased energy consumption, a smaller value of the soft threshold generates more accurate information of the network, thus users can control the trade-off between energy efficiency and data accuracy by the parameters adjustment. Moreover, the soft threshold can be varied and the users can change the fresh parameters as required at every cluster change time.

According to above discussion, TEEN has the following advantages: (1) Based on the two thresholds, data transmission can be controlled commendably, i.e., only the sensitive data demanded, can be transmitted, so that it reduces the energy transmission consumption and improves the effectiveness and usefulness of the receiving data; (2) TEEN is complement for reacting to large changes in the sensed attributes, which is suitable for reactive scenes and time critical applications. However, there exist a few drawbacks in TEEN as follows: (1) It is not suitable for periodic reports applications since the user may not get any data at all if the values of the attributes may not reach the threshold (Kandris et al 2009); (2) There exist wasted time-slots and a possibility that the BS may not be able to distinguish dead nodes from alive ones, because only when the data arrive at the hard threshold and has a variant higher than the soft threshold did the sensors report the data to the BS; (3) If CHs are not in the communication range of each other the data may be lost, because information propagation is accomplished only by CHs (Kandris et al 2011).
2.3 FIRST ORDER RADIO MODEL

Currently there is a great deal of research in the area of low energy radios. Different assumptions about the radio characteristics (Wang et al 1999), including energy dissipation in transmit and receive modes, will change the advantages of different protocols. The above work assumes a simple model where the radio dissipates $E_{\text{elec}} = 70 \text{ nJ/bit}$ to run the transmitter or receiver circuitry and $\varepsilon_{\text{amp}} = 120 \text{ pJ/bit/m}^2$ for the transmit amplifier to achieve an acceptable $E_b/N_0$. These parameters are slightly better than the current state-of-the-art in radio design. It also assumes a $d^2$ energy loss due to channel transmission (Heinzelman et al 2000), to transmit a $k$-bits message for a distance ‘$d$’ using the radio model shown in Figure 2.1, the energy spent is given in Equations 2.2 and 2.3 which are as follows:

$$E_{\text{TX}}(k,d) = E_{\text{elec}} \times k + E_{\text{amp}} \times k \times d^2 \quad (2.2)$$

and to receive this message, the radio expends:

$$E_{\text{RX}}(k) = E_{\text{elec}} \times k \quad (2.3)$$

Figure 2.1 First Order Radio Model
Here, $K$ represents the number of bits, $E_{\text{elec}}$ is the energy dissipated per bit to run the transmitter or the receiver circuit, and $E_{\text{amp}}$ depends on the transmitter amplifier model and $d$ is the transmission distance. For these parameter values, receiving a message is not a low cost operation; the protocol thus should try to minimize not only the transmit distances but also the number of transmit and receive operations for each message. The assumption made that the radio channel is symmetric such that the energy required transmitting a message from node A to node B is the same as the energy required transmitting a message from node B to node A for a given Signal to Noise Ratio (SNR). It also assumes that all sensors are sensing the environment at a fixed rate and thus always have data to send to the end user. For future versions of these protocols, it will implement an ‘event-driven’ simulation, where sensors only transmit data for some event occurs in the environment.

Micro sensor networks can contain hundreds or thousands of sensing nodes. It is desirable to make these nodes as cheap and energy-efficient as possible and rely on their large numbers to obtain high quality results. Network protocols must be designed to achieve fault tolerance in the presence of individual nodes failure while minimizing energy consumption. In addition, since the limited wireless channel bandwidth must be shared among all the sensors in the network, routing protocols for these networks should be able to perform local collaboration to reduce bandwidth requirements. Eventually, the data being sensed by the nodes in the network must be transmitted to a control center or base station, where the end-user can access the data. There are many possible models for these micro sensors networks. In this work, considered sensor networks are as follows:

- The base station is fixed and located far from the sensors.
• All nodes in the network are homogenous and energy constrained.

Thus, communication between the sensor nodes and the base station is expansive, and there are no ‘high-energy’ nodes through which communication can proceed.

2.4 DIRECT TRANSMISSION (DT)

Using a direct communication protocol, each sensor sends its data directly to the base station. If the base station is far away from the nodes, direct communication will require a large amount of transmit power from each node. This will quickly drain the battery of nodes and reduce the system lifetime. However the only reception in this protocol occur at the base station, so if either the base station is close to the nodes, or the energy required receiving data is large, this may be an acceptable (and possibly optimal) method of communication (Wang et al 1999).

2.5 META-HEURISTIC OPTIMIZATION METHODS

Hard optimization problems are defined as problems that cannot be solved to optimality, or to any guaranteed bound, by any exact (deterministic) method within a ‘reasonable’ time limit. These problems can be divided into several categories depending on whether they are continuous or discrete, constrained or unconstrained, mono or multi-objective, static or dynamic. In order to find satisfactory solutions for these problems, meta-heuristics can be used. A meta-heuristic is an algorithm designed to solve approximately a wide range of hard optimization problems without having to deeply adapt to each problem. Indeed, the greek prefix ‘meta’, present in the name, is used to indicate that these algorithms are ‘higher level’ heuristics, in contrast with problem-specific heuristics. Meta-heuristics are generally applied to a
problem for which there is no satisfactory problem-specific algorithm to solve them. They are widely used to solve complex problems in industry and services, in areas ranging from finance to production management and engineering. Almost all meta-heuristics share the following characteristics: they are nature-inspired (based on some principles from physics, biology or ethology); they make use of stochastic components (involving random variables); they do not use the gradient or Hessian matrix of the objective function; they have several parameters that need to be fitted to the problem at hand.

In the last thirty years, a great interest has been devoted to meta-heuristics. It is tried to point out some of the steps that have marked the history of meta-heuristics. One pioneer contribution is the proposition of the simulated annealing method by Kirkpatrick et al (1982). The Tabu search was proposed by Glover (1986), and the Artificial Immune System was proposed by Farmer et al. (1986). In 1988, Koza registered his first patent on genetic programming, later published in 1992. In 1989, Goldberg published a well known book on genetic algorithms. Dorigo (1992) completed his PhD thesis, in which he described his innovative work on ant colony optimization. The first algorithm based on bee colonies was proposed by Walker et al (1993). Another significant progress is the development of the particle swarm optimization by Kennedy & Eberhart (1995). The estimation of distribution algorithm proposed by Muhlenbein & PaarB (1996). Differential evolution proposed by storn & price (1997). Passino (2002) introduced an optimization algorithm based on bacterial foraging. Then, bio-geography-based optimization algorithm proposed by Simon(2008). The considerable development of meta-heuristics can be explained by the significant increase in the processing power of the computers, and by the development of massively parallel architectures. A meta-heuristic will be successful on a given optimization problem if it can provide a balance between the exploration
(diversification) and the exploitation (intensification). Exploitation is needed to identify parts of the search space with high quality solutions. Exploitation is important to intensify the search in some promising areas of the accumulated search experience. In this section, some popular and effective meta-heuristic optimization methods are analyzed.

2.5.1 Annealing Algorithm

This probabilistic meta-heuristic is inspired from physical process of annealing the crystalline materials. This process consists at heating a material at high temperature and then it must be slowly cooling to enhance its crystals' size. Atoms of heated material have a lofty energy that causes them to change positions and they can perform large random movements in the material. The slow cooling reduces atoms' energy and their movement capacity. The different cooling transitory states make it possible to obtain homogeneous materials with good quality. To implement this process by an optimization method, random movements of each point will be associated with a probability of a dependent variable representing the temperature of the material. Simulated annealing is based on the Metropolis algorithm (Metropolis et al 1953; Hastings 1970) that allows describing the evolution of a thermodynamic system. The link between this algorithm and optimization problems has been proposed for the first time by Pincus (1970), but Kirkpatrick et al (1983) and Cerny (1985), in a separate research works, are the pioneer of the developed form of simulated annealing algorithm. The simulated annealing algorithm becomes fast popular, owing to its easy adaptation to various problems and its efficiency. However, the principal disadvantage of this algorithm is the large parameters number (initial temperature, the temperature decrease rule, the temperature stages' duration, etc.), that makes it quite empirical. Another drawback is the slowness of this method. To overcome this problem, several Parallelization techniques have
been introduced Azencott (1992). Although initially created to be applied with discrete variables, the simulated annealing possesses continuous versions.

2.5.2 Tabu Search

Glover (1986) introduces Tabu search algorithm. This meta-heuristic is a mathematical optimization method used to solve combinatorial problems. This algorithm improves local search performance by adding to the research process a memory describing visited solutions. From a given position, Tabu search algorithm explores the neighborhood of this position and chooses a new one that optimizes the objective function. This search procedure is iteratively repeated until fixed criterions are satisfied. Thus every potential solution is marked as Tabu and added to the memory, also called Tabu list. In the next iterative the algorithm does not visit the stored solutions in the list. The main advantage of the Tabu search algorithm is having less parameter than simulated annealing algorithm.

2.5.3 Evolutionary Algorithms (EA)

Fraser (1957) is the pioneer of evolutionary algorithms. They represent a family of research algorithms inspired from species' biological evolution, such as: natural selection, mutation, reproduction, and recombination. Evolutionary algorithms idea is too simple. In a first step, a set of point called initial population is randomly built in a predefined search space. Each point or individual possesses a performance degree that measures its adaptation level to the target objective. An evolutionary algorithm evolves gradually, by successive iterations or generations, the population composition with maintaining its constant size (Selvakennedy et al 2007). The goal is the overall improvement of individuals’ performance through generations. In
each generation one applies a series of operators to population individuals that generate a new population. Each operator uses one or more population individuals, called parents, to generate new candidates, called offsprings. An evolutionary algorithm has usually three key operators: selection operator, crossover operator and mutation operator. Selection operator favors spread of better solutions in the population while maintaining its genetic diversity. The crossover operator is implemented during the creation phase of offsprings. This operator aims to exchange the parents’ genes to create offsprings. Mutation operator consists to draw randomly component of parent and replace it by a random value. This operator allows creating randomly offsprings and maintaining the population diversity. A complete list of all existing methods to define these operators is available in Eiben & Smith (2003).

### 2.5.4 Ant Colony Optimization (ACO) Algorithm

Ant Colony Optimization (ACO) algorithm are born from a simple observation. Insects, particularly ants, solve naturally complex problems. The principal factor that facilitates this behavior is that ants communicate with each other indirectly owing to deposit of chemicals substances called pheromones. This indirect communication type is called stigmergy. According to Goss et al (1989), if an obstacle is introduced in ants path, they will all tend, after a research phase, to follow the shortest way between nest and obstacle. They are more attracted to the area where the pheromone substance rate is highest. Ants that passed by food source and arrived too quickly to the nest are those that have taken the shorter way. Therefore, the pheromone quantity in this way will be more important than the longer distance. Thereby, eventually the shortest way has a greater probability to be used by all ants than other ways. First algorithms inspired from this analogy were proposed by Colorni et al (1992) and Dorigo et al (1996) to resolve problem of business traveler. In these algorithms, each solution is considered
as an ant moving in the search space. Ants mark better solutions and take account of previous markings to optimize their research. Ant colony algorithms use an implicit probability distribution to perform the transition between iterations. In their adapted version to combinatorial problems these algorithms use an iterative construction of solutions. These algorithms have been extended to resolve several discrete and continuous optimization problems, (Dorigo et al 2005). Ant colony algorithms have several advantages such as high intrinsic parallelism, robustness (a colony can maintain an effective search even if some of its individuals are defective) or decentralized (the ants do not obey a centralized authority).

2.5.5 Artificial Bee Colony (ABC) Algorithm

The Artificial Bee Colony (ABC) algorithm is a branch of nature inspired or swarm intelligence based meta-heuristic algorithm which was proposed by Karaboga & Basturk (2007) for optimizing numerical problems. It was motivated by the intelligent foraging behavior of honey bees. The algorithm is particularly based on the model proposed for the foraging manners of honey bee colonies. The model comprises three vital fundamentals: employed and unemployed foraging bees, and food sources (Zhu & Kwong 2010). The first two fundamentals, employed and unemployed foraging bees search for rich food sources, which is the third fundamental, is being close to their hive. The two principal modes of behavior are also described by the model, which are necessary for self-organization and collective intelligence: recruitment of foragers to the rich food sources resulting in positive feedback and abandonment of poor food sources by foragers causing negative feedback (Cheng-Jian & Shih-Chieh 2009).

In ABC, the colony consists of three groups of bees: employed bees linked with specific food sources, onlooker bees studying the dancing
behavior of employed bees in the hive to choose the desired food source and scout bees searching for food sources randomly once the employed is stuck with unsatisfactory food source. Both onlookers and scouts are also known as unemployed bees. The positions of all food sources are discovered by the scout bees originally. Thereafter, the exploitation of nectar of food sources are carried out by employed bees and onlooker bees. The repetitive exploitation will eventually cause them to become exhausted. Then, the employed bee which was exploiting the exhausted food source becomes a scout bee in search of other food sources once again. In other words, the employed bee whose food source has been exhausted becomes a scout bee. The position of a food source in ABC corresponds to the possible solution to the problem and the nectar amount of a food source signifies the quality (fitness) of the associated solution. The number of employed bees is equal to the number of food sources (solutions), since each employed bee is associated with one and only one food source.

ABC algorithm as a population-based meta-heuristic algorithm competes well with other population-based algorithms with an advantage of employing fewer control parameters (Karaboga & Akay 2009). Due to its simplicity and ease of implementation, the ABC algorithm has captured much attention and has been applied to solve many practical optimization problems such as structural and concrete analyses (Kang et al 2009), integer programming (Akay & Karaboga 2009), leaf-constrained minimum spanning tree (Singh 2009), digital IIR filter (Karaboga 2009), real parameter control, generalized assignment problem (Baykasoglu et al 2007), protein folding simulation, training of artificial neural networks (Karaboga & Akay 2007), Numerical optimization constrained optimization problems (Karaboga & Basturk 2007), cluster based wireless sensor network routing (Karaboga et al 2010), fuzzy clustering (Karaboga & Ozturk 2010), reconfiguring distribution
network (Linh & Anh 2010). The general format of the ABC algorithm proposed by Karaboga is as follows:

- Send the scouts onto the initial food sources

- REPEAT
  - Send the employed bees onto the food sources and determine their nectar amounts
  - Calculate the probability value of the sources with which they are preferred by the onlooker bees
  - Send the onlooker bees onto the food sources and determine their nectar amounts
  - Abandon the exploitation process, if the sources are exhausted by the bees
  - Send the scouts into the search area for discovering new food sources, randomly
  - Memorize the best food source found so far

- UNTIL (requirements are met)

This algorithm is used in many applications (Akay & Karaboga 2009) i.e., electrical power system, parallel and grid computing, data clustering and image analysis, computer science application, signal processing and communication.
2.5.6 Firefly Algorithm (FA)

Firefly algorithm proposed by Yang (2009) can be considered as an unconventional swarm-based heuristic algorithm for constrained optimization tasks inspired by the flashing behavior of fireflies. The algorithm constitutes a population-based iterative procedure with numerous agents (perceived as fireflies) concurrently solving a considered optimization problem. Agents communicate with each other via bioluminescent glowing which enables them to explore cost function space more effectively than in standard distributed random search. Intelligence optimization technique is based on the assumption that solution of an optimization problem can be perceived as agent (firefly) which glows proportionally to its quality in a considered problem setting. Consequently each brighter firefly attracts its partners (regardless of their sex), which makes the search space being explored more efficiently. The firefly algorithm has three particular idealized rules which are based on some of the basic flashing characteristics of real fireflies (Gandomi et al 2013). They are the following:

1) All fireflies are unisex and they will move towards more attractive and brighter ones regardless of their sex.

2) The degree of attractiveness of a firefly is proportional to its brightness. Also the brightness may decrease as the distance from the other fireflies increases due to the fact that the air absorbs light. If there is not a brighter or more attractive firefly than a particular one it will then move randomly.

3) The brightness or light intensity of a firefly is determined by the value of the objective function of a given problem.
Advantage: Mainly uses real random numbers and is based on the global communication among the swarm particles (i.e., the firefly), hence more effective in multi objective optimization.

2.5.6.1 FAlight intensity and brightness concept

In the firefly algorithm, there are two important issues: the variation of light intensity and formulation of the attractiveness. For simplicity, one can always assume that the attractiveness of a firefly is determined by its brightness, which in turn is associated with the encoded objective function. In the simplest case for maximum optimization problems, the brightness \( I \) of a firefly at a particular location \( x \) can be chosen as \( I(x) \propto f(x) \). However, the attractiveness \( \beta \) is relative; it should be seen in the eyes of the beholder or judged by the other fireflies. Thus, it will vary with distance \( r_{ij} \) between firefly \( i \) and firefly \( j \). In addition, light intensity decreases with the distance from its source, and light is also absorbed in the media, so the attractiveness should be allowed to vary with the degree of absorption (Xin- she Yang 2010). In the simplest form, the light intensity \( I(r) \) varies according to the inverse square law given in Equation 2.4.

\[
I(r) = \frac{I_s}{r^2} \tag{2.4}
\]

where, \( I_s \) is the intensity at the source and \( r \) is the distance. As a firefly’s attractiveness is proportional to the light intensity seen by adjacent fireflies, the attractiveness \( \beta \) of a firefly can now be defined using Equation 2.5.

\[
\beta = \beta_0 e^{-\gamma r} \tag{2.5}
\]
where, $\beta_0$ is the attractiveness at $r = 0$ and $\gamma$ is the light absorption coefficient. As it is often faster to calculate $\frac{1}{1+r^2}$ than an exponential function, the above function, if necessary, can conveniently be approximated using Equation 2.6.

$$\beta = \frac{\beta_0}{1 + \gamma r^2} \quad (2.6)$$

The distance ($r_{ij}$) between any two fireflies $i$ and $j$ at $x_i$ and $x_j$, respectively, is the Cartesian distance is given in Equation 2.7.

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \quad (2.7)$$

where, $x_{i,k}$ is the $k^{th}$ component of the spatial coordinate $x_i$ of the $i^{th}$ firefly. In 2-D case, Equation 2.8 gives it.

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (2.8)$$

The movement of a firefly ‘i’ is attracted to another more attractive (brighter) firefly ‘j’ is determined by Equation 2.9.

$$X_i = X_i + \beta_0 e^{-\gamma r_{ij}} (X_j - X_i) + \alpha \in_i \quad (2.9)$$

where the second term is due to attraction and third term is randomization with $\alpha$ being the randomization parameter, and $\in_i$ is a vector of random numbers drawn from a Gaussian distribution or uniform distribution. For example, the simplest form is $\in_i$ can be replaced by rand where rand is a random number generator uniformly distributed in [0, 1]. For most the implementation, $\beta_0 = 1$ and $\alpha \in [0, 1]$. It is worth pointing out that
Equation 2.7 is a random walk based towards the brighter fireflies. If $\beta = 0$, it becomes a simple random walk. Furthermore, the randomization term can easily be extended to other distributions such as Levy flights. The parameter $\gamma$ now characterizes the variation of the attractiveness, and its value is crucially important in determining the speed of the convergence and how the FA algorithm behaves.

2.5.7 Harmony Search Algorithm (HSA)

When listening to a beautiful piece of classical music, who has ever wondered if there is any connection between music and finding an optimal solution to a tough design problem such as the water distribution networks or other design problems in engineering. Now for the first time ever, scientists have found such an interesting connection by developing a new algorithm, called Harmony Search. Harmony Search (HS) was first developed by Geem et al (2001), though it is a relatively new meta-heuristic algorithm, its effectiveness and advantages have been demonstrated in various applications. Since its first appearance in 2001, it has been applied to solve many optimization problems including function optimization, engineering optimization, water distribution networks, groundwater modeling, energy-saving dispatch, truss design, vehicle routing, and others (Lee & Geem 2005).

Harmony search is a music-based meta-heuristic optimization algorithm. It was inspired by the observation that the aim of music is to search for a perfect state of harmony. This harmony in music is analogous to find the optimality in an optimization process. The search process in optimization can be compared to a jazz musician’s improvisation process. On the one hand, the perfectly pleasing harmony is determined by the audio aesthetic standard. A musician always intends to produce a piece of music with perfect harmony. On the other hand, an optimal solution to an optimization problem should be
the best solution available to the problem under the given objectives and limited by constraints. Both processes intend to produce the best or optimum. Such similarities between two processes can be used to develop new algorithms by learning from each other. Harmony Search is just such a successful example by transforming the qualitative improvisation process into some quantitative rules by idealization, and thus turning the beauty and harmony of music into an optimization procedure through search for a perfect harmony, namely, the Harmony Search (HS) or Harmony Search algorithm.

For example, Consider a jazz trio composed of saxophone, double bass and guitar. There exist certain amount of preferable pitches in each musician’s memory: saxophonist, \{Do, Mi, Sol\}; double bassist, \{Si, Sol, Re\}; and guitarist, \{La, Fa, Do\}. If saxophonist randomly plays \{Sol\} out of \{Do, Mi, Sol\}, double bassist \{Si\} out of \{Si, Sol, Re\}, and guitarist \{Do\} out of \{La, Fa, Do\}, that harmony (Sol, Si, Do) makes another harmony (musically C-7 chord). And if the New Harmony is better than existing worst harmony in the HM, the New Harmony is included in the HM and the worst harmony is excluded from the HM. This procedure is repeated until fantastic harmony is found.

2.5.8 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is an artificial intelligence mechanism, which is motivated by the social behaviors of natural species, for instance-herd of animals, swarm of birds, etc. (Kulkarni & Venayagamoorthy 2011). The population-based swarm intelligence method executes with the aim of optimizing an objective (or fitness) function. The algorithm employs a swarm of search points (also called particles) and tracks the fitness of each particle. Every particle is associated with corresponding velocity. This assists the particle to move onto a better position, if the cost to the objective function
is optimized. PSO is known to perform better than other swarm intelligence techniques (Ant Colony Optimization, Genetic Algorithm, etc.) in terms of computation complexity and convergence rate (Djeffal et al 2009), (Katari et al 2008). In PSO, the particles possess a fitness criterion for all the positions they visit. Hence, it becomes essential to maintain a local best fitness value for the particles during every generation (iteration). The local information further helps to locate the trajectory towards global best position. The pattern emerged from the collective intelligence of the particles ultimately optimizes the objective function. In this section, the swarm optimization is applied to find cluster head positions in order to reduce the intra-cluster distance and overall energy consumption during packet transmission to the sink (Chatterjee & Siarry 2006).

2.5.9 **Shuffled Frog Leap Algorithm (SFLA)**

Shuffled Frog Leaping Algorithm (SFLA) proposed by Eusuff & Lansey (2003). SFLA is a population-based cooperative meta-heuristic algorithm with efficient mathematical function and global search capability. The SFLA is a search metaphor inspired by natural memetics and evolution. It is inspired by the interactive behavior and global exchange of information of frogs searching for food laid on discrete stones randomly located in a pond (Yinghai et al 2010). It combines the advantages of the genetic-based Memetic algorithm (MA) and the social behavior-based PSO algorithm with such characteristics as simple concept, fewer parameters adjustment, prompt formation, great capability in global search and easy implementation. The steps in SFLA include the following:

- **Initial population:** Individual frogs are equivalent to the GA chromosomes, and represent a set of solutions
• **Sorting and distribution**: Frogs are sorted in descending order based on their fitness values, then each frog is distributed to a different subset of the whole population called a memeplex, the entire population is divided into m memeplexes, each containing n frogs.

• **Memeplex evolution**: An independent local search is conducted for each frog memeplex, in what is called memeplex evolution.

• **Shuffling**: After a defined number of memetic evolutionary steps, frogs are shuffled among memeplexes, enabling frogs to interchange messages among different memeplexes and ensure that they move to an optimal position, similar to particles in PSO.

• **Terminal condition**: If a global solution or a fixed iteration number is reached, the algorithm stops.

### 2.5.10 Bacterial Foraging Optimization (BFO) Algorithm

This has been evolving as a new and promising branch in Bio inspired Algorithms that can bridge the gap between microbiology and engineering. These classes of algorithms inherit the characteristics of bacterial foraging patterns such as chemo taxis, metabolism, reproduction and quorum sensing. The complex and organized activities exhibited in bacterial foraging patterns inspire a new approach to solve complex optimization problems. The Bacterial Foraging Optimization (BFO) Algorithm was introduced by Passino (2002). Foraging is a phenomenon of a bacterial colony rather than an individual behavior. BFO Algorithm consists of three principal mechanisms namely, chemo taxis, reproduction, and elimination-dispersal.
Chemotaxis (cell movement) is the activity of bacteria gathering to nutrient-rich areas in a spontaneous fashion; in this context, a cell-to-cell communication mechanism is established to simulate the biological behavior of bacterial movement (swim/tumble).

Reproduction comes from the concept of natural selection; under this procedure, only the best adapted bacteria tend to survive and transmit their genetic characters to succeeding generations, while the less adapted ones tend to perish.

Elimination-dispersal events randomly select parts of the bacteria population to diminish and disperse into random positions in the environment; this way the algorithm ensures the diversity of the species, and prevents getting trapped to local optima, improving global search ability.

Bacterial Foraging Optimization (BFO) is a population-based numerical optimization algorithm. In recent years, bacterial foraging behavior has provided rich source of solution in many engineering applications and computational model. It has been applied for solving practical engineering problems like optimal control, harmonic estimation, channel equalization etc. In this thesis, BFO has been used for cluster head selection to provide improved energy efficiency in routing.

2.6 SUMMARY

Energy constraints are the big challenge in wireless sensor network design. In this chapter, the various issues related to energy management and the existing energy efficient clustering protocols were discussed with their merits and demerits. Several optimization algorithms have been developed for
solving combinatorial and numeric optimization problems. These algorithms can be classified into different groups depending on the criteria being considered, such as population based, iterative based, stochastic, deterministic, etc. While an algorithm working with a set of solutions and trying to improve them is called population based, the one using multiple iterations to approach the solution sought is named as iterative algorithm. If an algorithm employs a probabilistic rule for improving a solution then it is called probabilistic or stochastic. This thesis explains meta-heuristic based optimization algorithm method that can be used in WSN for finding the optimal cluster heads which effectively improves the lifetime of WSN.