Computational Intelligence in Wireless Sensor Networks: A Survey

Raghavendra V. Kulkarni, *Senior Member, IEEE,* Anna Förster, *Member, IEEE* and
Ganesh Kumar Venayagamoorthy, *Senior Member, IEEE*

***Abstract*—Wireless sensor networks (WSNs) are networks of distributed autonomous devices that can sense or monitor physical or environmental conditions cooperatively. WSNs face many challenges, mainly caused by communication failures, storage and computational constraints and limited power supply. Paradigms of computational intelligence (CI) have been successfully used in recent years to address various challenges such as data aggregation and fusion, energy aware routing, task scheduling, security, optimal deployment and localization. CI provides adaptive mechanisms that exhibit intelligent behavior in complex and dynamic environments like WSNs. CI brings about flexibility, autonomous behavior, and robustness against topology changes, communication failures and scenario changes. However, WSN developers are usually not or not completely aware of the potential CI algorithms offer. On the other side, CI researchers are not familiar with all real problems and subtle requirements of WSNs. This mismatch makes collaboration and development difficult. This paper intends to close this gap and foster collaboration by offering a detailed introduction to WSNs and their properties. An extensive survey of CI applications to various problems in WSNs from various research areas and publication venues is presented in the paper. Besides, a discussion on advantages and disadvantages of CI algorithms over traditional WSN solutions is offered. In addition, a general evaluation of CI algorithms is presented, which will serve as a guide for using CI algorithms for WSNs.**

***Index Terms*—clustering, computational intelligence, data aggregation, deployment, design, localization, quality of service, routing, scheduling, security, sensor fusion, wireless sensor networks**

List of Abbreviations

ACO	Ant colony optimization
AIS	Artificial immune system
CI	Computational intelligence
CSMA	Carrier sense multiple access
DE	Differential evolution
EA	Evolutionary algorithm
GA	Genetic algorithm
LEACH	Low energy adaptive clustering hierarchy
MAC	Media Access Control
MANET	Mobile ad hoc network
NN	Neural network
PSO	Particle swarm optimization
QoS	Quality of Service Management
RL	Reinforcement learning
RSSI	Received signal strength indication
RTS	Request to send
SI	Swarm intelligence
SOM	Self-organizing map
TDMA	Time division multiple access
TPOT	Team-partitioned, opaque-transition
WSN	Wireless sensor network

Authors acknowledge the support received for this work from the National Science Foundation, USA, under the grants ECCS # 0625737 and ECCE # 0348221, and National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant # 5005-67322.

Raghavendra V. Kulkarni and Ganesh Kumar Venayagamoorthy are with the Real-Time Power and Intelligent Systems Laboratory, Missouri University of Science and Technology, Rolla, MO-65409, USA e-mail: ({arvie,gkumar}@ieee.org).

Anna Förster is with the Faculty of Informatics, University of Lugano, Switzerland e-mail: (anna.foerster@ieee.org).

Manuscript received January 07, 2009; revised October 23, 2009.

I. Introduction

A Wireless sensor network is a network of distributed autonomous devices that can sense or monitor physical or environmental conditions cooperatively [1]. WSNs are used in numerous applications such as environmental monitoring, habitat monitoring, prediction and detection of natural calamities, medical monitoring and structural health monitoring [2]. WSNs consist of a large number of small, inexpensive, disposable and autonomous sensor nodes that are generally deployed in an ad hoc manner in vast geographical areas for remote operations. Sensor nodes are severely constrained in terms of storage resources, computational capabilities, communication bandwidth and power supply. Typically, sensor nodes are grouped in clusters, and each cluster has a node that acts as the cluster head. All nodes forward their sensor data to the cluster head, which in turn routes it to a specialized node called sink node (or base station) through a multi-hop wireless communication as shown in Figure 1. However, very often the sensor network is rather small and consists of a single cluster with a single base station [3]–[5]. Other scenarios such as multiple base stations or mobile nodes are also possible. Article [6] presents a classification of WSNs based on communication functions, data delivery models, and network dynamics. Resource constraints and dynamic topology pose technical challenges in network discovery, network control and routing, collaborative information processing, querying, and tasking [2]. CI combines elements of learning, adaptation, evolution and fuzzy logic to create intelligent machines. In addition to paradigms like neuro-computing, reinforcement learning, evolutionary computing and fuzzy computing, CI encompasses techniques that use swarm intelligence, artificial immune systems and hybrids of two or more of the above. Paradigms of

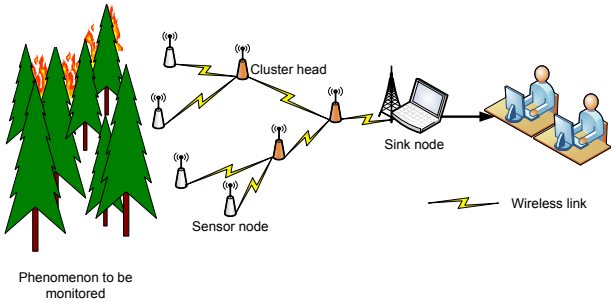

Fig. 1. Architecture of a typical wireless sensor network

CI have found practical applications in areas such as product design, robotics, intelligent control, biometrics and sensor networks. Researchers have successfully used CI techniques to address many challenges in WSNs. However, various research communities are developing these applications concurrently, and a single overview thereof does not exist. Most of the here presented works have scattered in journals and conferences whose prime focus is not WSNs. Aim of this survey is to bridge this gap and present a brief, but comprehensive survey of numerous CI approaches and applications, which provide the WSN researchers with new ideas and incentives. A discussion on yet unexplored challenges in WSNs, and a projection on potential CI applications in WSN are presented with an objective of encouraging researchers to use CI techniques in WSN applications.

The rest of this paper is organized as follows: The main challenges in WSNs are discussed in Section II. The paradigms of CI are outlined in Section III. The WSN applications that use CI approaches are clustered in three groups and discussed in Sections IV, V and VI. A high level evaluation of CI techniques in terms of their applicability to WSNs is provided in Section VII, wrapped as a guide for WSN application developers and practitioners. Finally, conclusions and authors' vision on future trends and directions of applying CI methods to various WSN problems are laid out in Section VIII.

II. Challenges in Sensor Networks

Real deployments of wireless sensor networks usually implement one of the three general applications: periodic reporting, event detection, and database-like storage. Periodic reporting is by far the most used and the simplest application scenario, in which at regular intervals the sensors sample their environment, store the sensory data, and send it to the base station(s). Actuators such as automatic irrigation systems and alarm systems are often connected with such WSNs. This scenario is used in most monitoring applications in agriculture [7], [8], microclimate [4], [5], [9] and habitat surveillance [10]–[12], military operations [13], and disaster relief [14]. The main property of periodic reporting applications is the predictability of the data traffic and volume.

In contrast, in event detection applications [3], [15], nodes sense the environment and evaluate the data immediately for its usefulness. If useful data (an event) is detected, the data is transmitted to the base station(s). The data traffic can hardly be predicted: events usually occur randomly and the resulting data traffic is sporadic. However, a small amount of data has to be exchanged for route management and aliveness checks even when no events are detected.

The third group of sensor networks applications, database-like storage systems [16], are similar to event-based systems. All sensory data (regular sampling or events) is stored locally on the nodes. Base stations search for interesting data and retrieve it from the nodes directly. The main challenge in these applications is to store the data in a smart way for fast search and retrieval.

The challenges and properties of WSN deployments can be summarized as follows:

Wireless ad hoc nature: A fixed communication infrastructure does not exist. The shared wireless medium places additional restrictions on the communication between the nodes and poses new problems like unreliable and asymmetric links. But, it provides the broadcast advantage: A packet transmitted by a node to another is received by all neighbors of the transmitting node.

Mobility and topology changes: WSNs may involve dynamic scenarios. New nodes may join the network, and the existing nodes may either move through the network or out of it. Nodes may cease to function, and surviving nodes may go in or out of transmission radii of other nodes. WSN applications have to be robust against node failure and dynamic topology.

Energy limitations: Nodes in most WSNs have limited energy. The basic scenario includes a topology of sensor nodes, and a limited number of more powerful base stations. Maintenance or recharging of the batteries on sensor nodes is not possible after deployment. Communication tasks consume maximum power available to sensor nodes, and in order to ensure sustained long-term sensing operation, communication tasks need to be exercised frugally.

Physical distribution: Each node in a WSN is an autonomous computational unit that communicates with its neighbors via messages. Data is distributed throughout the nodes in the network and can be gathered at a central station only with high communication costs. Consequently, algorithms that require global information from the entire network become very expensive. Thus, reticent distributed algorithms are highly desirable.

A brief description of the major WSN challenges addressed by CI techniques is presented in the following subsections:

A. Design and Deployment

WSNs are used in vastly diversified applications ranging from monitoring a biological system through tissue implanted sensors to monitoring forest fire through air-dropped sensors. In some applications, the sensor nodes need to be placed accurately at predetermined locations, whereas in others, such positioning is unnecessary or impractical. Sensor network design aims at determining the type, amount and location of sensor nodes to be placed in an environment in order to get a complete knowledge of its functioning conditions.

B. Localization

Node localization refers to creating location awareness in all deployed sensor nodes. Location information is used to detect

and record events, or to route packets using geometric-aware routing [17], [18]. Besides, location itself is often the data that needs to be sensed. An overview of localization systems for WSNs is presented in [19]. Localization methods that use time-of-arrival of signals from multiple base stations are commonly used in WSNs [20].

C. Data Aggregation and Sensor Fusion

Sensor fusion is the process of combining of the data derived from multiple sources such that either the resulting information is in some sense better than would be possible with individual sources, or the communication overhead of sending individual sensor readings to the base station is reduced. Due to large-scale deployment of sensors, voluminous data is generated, efficient collection of which is a critical issue. Most widely used non-CI methods for sensor fusion include Kalman filter, Bayesian networks and Dempster-Shafer method [21]. A survey of data aggregation algorithms used in WSNs is presented in [22].

D. Energy Aware Routing and Clustering

Economic usage of energy is important in WSNs because replacing or recharging the batteries on the nodes may be impractical, expensive or dangerous. In many applications, network life expectancy of a few months or years is desired.

Routing refers to determining a path for a message from a source node to a destination node. In proactive routing methods, routing tables are created and stored regardless of when the routes are used. In reactive routing methods, routes are computed as necessary. In densely deployed networks, routing tables take a huge amount of memory, and therefore, hybrids of proactive and reactive methods are suitable for such networks. Another possible solution is to cluster the network into hierarchies. An overview of modern WSN routing algorithms is presented in [23].

E. Scheduling

In order to conserve energy, typical sensor nodes remain in sleep mode most of the time, and go into active mode periodically in order to acquire and transmit sensory data. A strict schedule needs to be followed regarding when a node should wake up, sense, transmit (or perform locomotion), ensuring maximum network lifetime. Causing the WSN nodes to take right actions at right time is the major objective of WSN scheduling.

F. Security

Wireless links in WSNs are susceptible to eavesdropping, impersonating, message distorting etc. Poorly protected nodes that move into hostile environments can be easily compromised. Administration becomes more difficult due to dynamic topology. Various security challenges in wireless sensor networks are analyzed and key issues that need to be resolved for achieving adequate security are summarized in [24]. Types of routing attacks and their countermeasures are presented in [25]. A review of security threats to WSNs and a survey of defense mechanisms is presented in [26].

G. Quality of Service Management

QoS is an overused term that has various meanings and perspectives. QoS generally refers to the quality as perceived by the user/application, while in the networking community, QoS is accepted as a measure of the service quality that the network offers to the applications/users. QoS refers to an assurance by the network to provide a set of measurable service attributes to the end-to-end users/applications in terms of fairness, delay, jitter, available bandwidth, and packet loss. A network has to provide the QoS while maximizing network resource utilization. To achieve this goal, the network is required to analyze the application requirements and deploy various network QoS mechanisms. A survey of QoS support in wireless sensor networks is presented in [27].

III. A Brief Overview of the Paradigms of CI

CI is the study of adaptive mechanisms that enable or facilitate intelligent behavior in complex and changing environments [28], [29]. These mechanisms include paradigms that exhibit an ability to learn or adapt to new situations, to generalize, abstract, discover and associate. In [30], CI is defined as *the computational models and tools of intelligence capable of inputting raw numerical sensory data directly, processing them by exploiting the representational parallelism and pipelining the problem, generating reliable and timely responses and withstanding high fault tolerance*. Paradigms of CI are designed to model the aspects of biological intelligence. CI encompasses paradigms such as neural networks, reinforcement learning, swarm intelligence, evolutionary algorithms, fuzzy logic and artificial immune systems. These paradigms are briefly introduced in the following subsections. Besides, it is common to find hybrids of these paradigms, such as neuro-fuzzy systems, fuzzy-immune systems etc. Certainly there exist more CI techniques, which are not discussed here because they have not been applied to WSNs problems yet and their properties do not suit the requirements well. A broader discussion of CI can found in [29], [30].

A. Neural Networks

The human brain, which possesses an extraordinary ability to learn, memorize and generalize, is a dense network of over 10 billion neurons, each connected on average to about 10,000 other neurons. Each neuron receives signals through synapses, which control the effects of the signals on the neuron. These synaptic connections play an important role in the behavior of the brain. These findings have inspired modeling of biological neural systems by means of NNs [31]. The three basic components of an artificial neuron shown in Figure 2 are:

1) The links that provide weights W_{ji}, to the n inputs of j^{th} neuron x_i, $i = 1, \ldots, n$;
2) An aggregation function that sums the weighted inputs to compute the input to the activation function $u_j = \Theta_j + \sum_{i=1}^{n} x_i W_{ji}$, where Θ_j is the bias, which is a numerical value associated with the neuron. It is convenient to

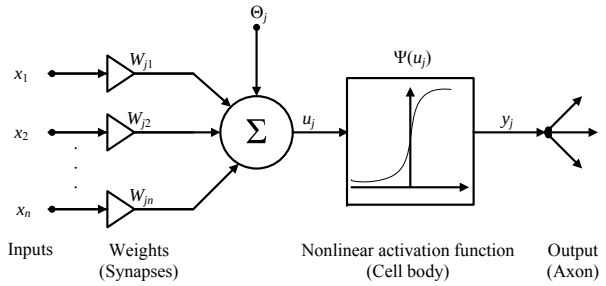

Fig. 2. Structure of an artificial neuron

think of the bias as the weight for an input x_0 whose value is always equal to one, so that $u_j = \sum_{i=0}^{n} x_i W_{ji}$;

3) An activation function Ψ that maps u_j to $v_j = \Psi(u_j)$, the output value of the neuron. Some examples of the activation functions are: step, sigmoid, tan hyperbolic and Gaussian function.

An NN consists of a network of neurons organized in input, hidden and output layers. In feedforward NNs, the outputs of a layer are connected as the inputs to the next layer while in recurrent networks, feedback connections are allowed as shown in dotted lines in Figure 3. In an Elman type recurrent network, a copy of hidden layer output, referred to as the *context layer*, is presented as the input to the hidden layer. In a Jordan type recurrent network, a copy of output of the neurons in the output layer is presented as the input to the hidden layer. Following is a non-exhaustive list of the types of NNs that are found in literature.

- single-layer networks (e.g., Hopfield network);
- multilayered feedforward networks (e.g., Perceptron, generalized neuron);
- temporal networks (e.g., Elman network, Jordan network);
- self-organizing networks (e.g., Kohonen's map).

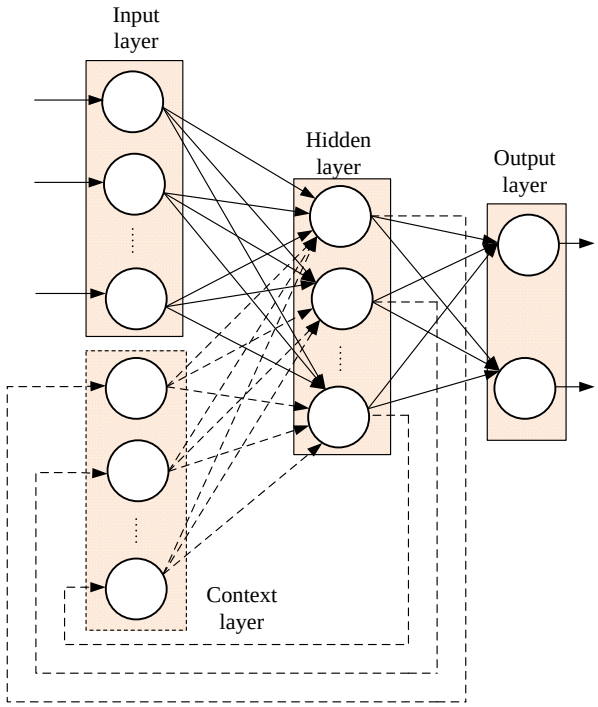

Fig. 3. Popular NN architectures: The connections shown in solid lines and the context later make up a feedforward NN. Addition of the connections shown in dotted lines converts it into a recurrent neural network.

NNs learn the facts represented by patterns and determine their inter-relationships. Learning is the process in which the weights of an NN are updated in order to discover patterns or features in the input data. Learning methods are classified into the following types: i) supervised learning, ii) unsupervised learning and iii) reinforcement learning. In supervised learning, a teacher presents an input pattern and the corresponding target output. Network weights are adapted in such a way that the error is minimized. The objective of unsupervised learning is to discover patterns in the input data with no help from a teacher. In reinforcement learning, the learning agent communicates with its own environment, but not with a teacher. The goal is to find a *policy*, which selects an action at any time-step, which leads to the best possible *reward* from the environment. To each action of the agent the environment responds with a reward, which represents the effectiveness of the action in that time-step; however, there are no "correct" or "incorrect" actions. A survey of NN learning methods, their properties and applications is presented in [32]. NNs have been found successful in a wide range of applications such as power system stabilization, pattern classification, speech recognition, robotics, prediction and image processing.

B. Fuzzy logic

Classical set theory allows elements to be either included in a set or not. This is in contrast with human reasoning, which includes a measure of imprecision or uncertainty, which is marked by the use of linguistic variables such as *most, many, frequently, seldom* etc. This approximate reasoning is modeled by fuzzy logic, which is a multivalued logic that allows intermediate values to be defined between conventional threshold values. Fuzzy systems allow the use of fuzzy sets to draw conclusions and to make decisions. Fuzzy sets differ from classical sets in that they allow an object to be a partial member of a set. For example, a person may be a member of the set *tall* to a degree of 0.8 [33]. In fuzzy systems, the dynamic behavior of a system is characterized by a set of linguistic fuzzy rules based on the knowledge of a human expert. Fuzzy rules are of the general form: *if* antecedent(s) *then* consequent(s), where antecedents and consequents are propositions containing linguistic variables. Antecedents of a fuzzy rule form a combination of fuzzy sets through the use of logic operations. Thus, fuzzy sets and fuzzy rules together form the knowledge base of a rule-based inference system as shown in Figure 4.

Antecedents and consequents of a fuzzy rule form fuzzy *input space* and fuzzy *output* space respectively, which are defined by combinations of fuzzy sets. Non-fuzzy inputs

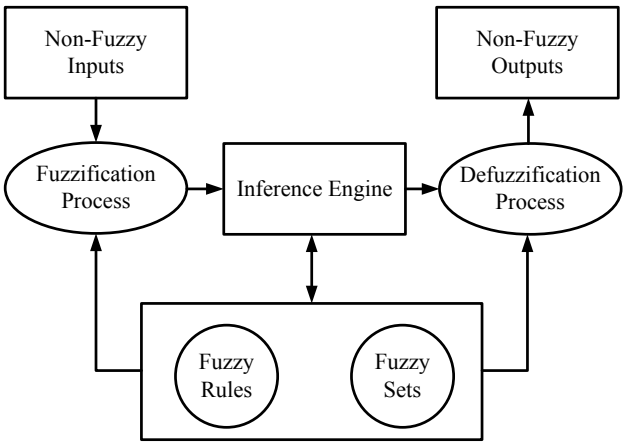

Fig. 4. Block diagram of a fuzzy inference system

are mapped to their fuzzy representation in the process called fuzzification. This involves application of membership functions such as triangular, trapezoidal, Gaussian etc. The inference process maps fuzzified inputs to the rule base to produce a fuzzy output. A consequent of the rule, and its membership to the output sets are determined here. The defuzzification process converts the output of a fuzzy rule into a crisp, non-fuzzy form. Popular inference methods that determine an approximate non-fuzzy scalar value to represent the action to be taken include max-min method, averaging method, root sum squared method, and clipped center of gravity method [34].

Fuzzy logic has been applied successfully in control systems (e.g., control of vehicle subsystem, power systems, home appliances, elevators etc.), digital image processing and pattern recognition.

C. Evolutionary Algorithms

Evolutionary algorithms model the natural evolution, which is the process of adaptation with the aim of improving survival capabilities through processes such as natural selection, survival-of-the-fittest, reproduction, mutation, competition and symbiosis. EC encompasses a variety of EAs that all share a common underlying idea of survival of the fittest. EAs use a population of solution candidates called chromosomes. Chromosomes are composed of genes which represent a distinct characteristic. A fitness function, which the EA seeks to maximize over the generations, quantifies the fitness of an individual chromosome. Process of reproduction is used to mix characteristics of two or more chromosomes (called parents) into the new ones (called offspring). Offspring chromosomes are mutated through small, random genetic changes in order to increase diversity. The fittest chromosomes are selected to go into the next generation, and the rest are eliminated. The process is repeated generation after generation until either a fit enough solution is found or a previously set computational limit is reached.

Following are the major classes of EAs.

- Genetic algorithms, which model genetic evolution
- Genetic programming whose individual chromosomes are computer programs
- Evolutionary programming which model adaptive behavior in evolution
- Evolutionary strategies which model strategy parameters that control variation in evolution
- Differential evolution which is identical to GA except for the reproduction mechanism
- Cultural evolution which models the evolution of culture of a population and culture's influence on genetic and adaptive evolution of individuals
- Coevolution in which initially "dumb" individual evolve through cooperation or competition and become fit enough to survive

Successful applications of EA include planning, design, control, classification and clustering, time series modeling, music composing etc.

D. Swarm Intelligence

SI originated from the study of collective behavior of societies of biological species such as flocks of birds, shoals of fish and colonies of ants. SI is the property of a system whereby collective behaviors of unsophisticated agents interacting locally with their environment cause coherent functional global patterns to emerge. While graceful but unpredictable bird-flock choreography inspired the development of particle swarm optimization [35], impressive ability of a colony of ants to find shortest path to their food inspired the development of ant colony optimization [36]. The honey bee algorithm mimics foraging behavior of swarms of honey bees [37].

1) Particle Swarm Optimization: The basic PSO consists of a population (or a swarm) of s particles, each of which represents a candidate solution [35]. The particles explore an n dimensional space in search of the global solution, where n represents the number of parameters to be optimized. Each particle i occupies position x_{id} and moves with a velocity v_{id}, $1 \leq i \leq s$ and $1 \leq d \leq n$. The particles are initially assigned random positions and velocities within fixed boundaries, i.e., $x_{\min} \leq x_{id} \leq x_{\max}$ and $v_{\min} \leq v_{id} \leq v_{\max}$ (in most cases $v_{\min} = -v_{\max}$). Fitness of a particle is determined from its position. The fitness is defined in such a way that a particle closer to the solution has higher fitness value than a particle that is far away. In each iteration, velocities and positions of all particles are updated to persuade them to achieve better fitness. The process of updating is repeated iteratively either until a particle reaches the global solution within permissible tolerance limits, or until a sufficiently large number of iterations is reached. Magnitude and direction of movement of a particle is influenced by its previous velocity, its experience and the knowledge it acquires from the swarm through social interaction.

In the $gbest$ version of PSO, each particle has a memory to store $pbest_{id}$, the position where it had the highest fitness. Besides, each particle can access the position $gbest_d$, the position of the particle having the maximum fitness. The $gbest$ particle represents the best solution found as yet. At each iteration k, PSO adds velocity v_{id} to each position x_{id} and

steers the particles towards its $pbest_{id}$ and $gbest_d$ using (1) and (2).

$$\begin{aligned} v_{id}(k+1) &= w \cdot v_{id}(k) + c_1 \cdot rand_1 \cdot (pbest_{id} - x_{id}) \\ &\quad + c_2 \cdot rand_2 \cdot (gbest_d - x_{id}) \quad (1) \\ x_{id}(k+1) &= x_{id}(k) + v_{id}(k+1) \quad (2) \end{aligned}$$

Here, $rand_1$ and $rand_2$ are random numbers having uniform distribution in the range (0, 1). The velocity update equation (1) shows that a particle is influenced by 3 components of acceleration. The first term involves the inertia coefficient w, $0.2 < w < 1.2$, and it denotes the inertia of the particle [38]. The second term involves the cognitive acceleration constant c_1. This component propels the particle towards the position where it had the highest fitness. The third term involves the social acceleration constant c_2. This component steers the particle towards the particle that currently has the highest fitness.

The velocity of a particle is bounded between properly chosen limits $v_{\max}$ and $v_{\min}$. Similarly, the position of a particle is restricted between properly chosen constants $x_{\max}$ and $x_{\min}$. Several variants of PSO have been devised [39] and applied to optimization problems in power systems, stock markets, antenna control and WSNs.

2) Ant Colony Optimization: ACO was introduced in [36] as a metaheuristic for solving combinatorial optimization problems. Foraging ants initially explore surroundings of their nest in a random manner. When an ant finds a source of food, it evaluates quantity and quality of the food and carries some food to the nest. While returning to the nest, the ant deposits a trail of chemical pheromone, which guides other ants to the food source. This characteristic of ant colonies is exploited in artificial ant colonies to solve combinatorial optimization problems. Consider that two paths A and B exist between a nest and a food source, and $n_A(t)$ and $n_B(t)$ number of ants use them at time step t respectively, then the probability of ant choosing path A at the time step $t+1$ is given by (3).

$$P_A(t+1) = \frac{(c + n_A(t))^\alpha}{(c + n_A(t))^\alpha + (c + n_B(t))^\alpha} = 1 - P_B(t+1) \quad (3)$$

where c is the degree of attraction of an unexplored branch, and α is the bias to using pheromone deposits in the decision process. An ant chooses between the path A or path B using the decision rule: if $U(0,1) \leq P_A(t+1)$ then choose path A otherwise choose path B.

The main idea of the ACO metaheuristic is to model the problem as a search for the best path in a "construction graph" that represents the states of the problem. Artificial ants walk through this graph, looking for good paths. They communicate by laying pheromone trails on edges of the graph, and they choose their path with respect to probabilities that depend on the amount of pheromone previously left.

In ACO, an artificial ant builds a solution by traversing the fully connected construction graph $G = (\vec{V}, \vec{E})$, where $\vec{V}$ is a set of vertices and $\vec{E}$ is a set of edges. Solution components may be represented either by vertices or by edges. Artificial ants move from a vertex to another along the edges of the graph, incrementally building a partial solution. Additionally, ants deposit a certain amount of pheromone on the components that they traverse. The amount of pheromone deposited $\Delta\tau$ may depend on the quality of the solution found. Subsequent ants use the pheromone information as a guide toward promising regions of the search space.

In Simple ACO, a small random amount of pheromone $\tau_{i,j}(0)$ is assigned initially to each edge (i, j). An ant randomly selects an edge to follow. A number of ants, $k = 1, \ldots, n_k$, are placed on the source node. In each iteration, each ant constructs a path to the destination node. An ant k at node i chooses the node $j \in N_i^k$ to visit next based on probability computed using (4), where N_i^k is the set of nodes connected to node i.

$$p_{ij}^k(t) = \begin{cases} \frac{\tau_{ij}^\alpha(t)}{\sum_{j \in N_i^k} \tau_{ij}^\alpha(t)} & \text{if } j \in N_i^k \\ 0 & \text{if } j \notin N_i^k \end{cases} \quad (4)$$

For an ant k at node i, if N_i^k is null, then the predecessor to node i is included in N_i^k. This causes the constructed paths to contain loops. These loops are eliminated when an ant reaches its destination node. When all the ants reach the desired destination node, each ant returns to the source in the reverse path and deposits a pheromone quantity,

$$\Delta\tau_{ij}^k(t) \propto \frac{1}{L^k(t)} \quad (5)$$

$L^k(t)$ is the length of the path constructed by ant k in time step t. The deposited pheromone $\Delta\tau_{ij}^k(t)$ determines the quality of the corresponding solution. In simple ACO, the quality of the solution is expressed as the reciprocal of the number of hops in the path. However, other measures such as cost of traveling or physical distance of the path can be used instead. The algorithm is terminated when a maximum number of iterations are done, an acceptable solution is found or all ants follow the same path.

ACO has been successfully applied to combinatorial optimization problems such as the traveling salesman problem, assignment problems, scheduling problems, and vehicle routing problems.

E. Artificial Immune Systems

The biological immune system defends the body from foreign pathogens. The three layers of a biological immune system are: *anatomic barrier, innate (nonspecific) immunity* and *adaptive (specific) immunity*, of which innate and adaptive immunities influence each other. When adaptive immunity detects the presence of a pathogen, it triggers a sequence of responses of *humoral* immunity and *cell-mediated* immunity. Innate immunity is directed on any pathogen. If the pathogen survives the innate immunity, the body directs the defence against a particular known type of pathogen. The biological immune system is a parallel, distributed adaptive system having decentralized control mechanism. This complex, adaptive biological immune system is modeled by a number of AIS models. These models capture such mechanisms of biological immune systems as negative selection, immune network model and clonal selection [40].

1) Immune network models: Immune network models are based on the hypothesis that the immune system uses an idiotypic network of interconnected B cells - a component of the adaptive immune system- for antigen recognition. The strength of interconnection between two B cells is proportional to the affinity they share. A B cell population has two types of sub-populations: the initial population and the cloned population. The initial subset is generated from raw training data, and the rest are used as antigen training items. Antigens are randomly selected from training set and presented to areas of the B cell network. In case of successful binding, the B cell is cloned and mutated, leading to a diverse set of antibodies. When a new B cell is created, it is integrated into the network at the closest B cells. If the new cell can not be integrated, it is removed from the population. If all binds fail, a B cell is generated using the antigen as a template and is then incorporated into the network.

2) Clonal selection: The clonal selection principal describes an immune response to an antigen stimulus. It postulates that the cells that can recognize the antigen proliferate. The main features of the clonal selection theory are:

- The new cells are clones of their parents with a mutation mechanism.
- Elimination of newly differentiated lymphocytes carrying self-reactive receptors.
- Proliferation and differentiation on contact of mature cells with antigens.

The CLONALG algorithm is based on clonal selection. It is similar to the evolutionary algorithms and has following features:

1) dynamical population,
2) exploration and exploitation of the search space,
3) ability to locate multiple optima,
4) ability to maintain local optimal solutions,
5) well defined stopping criterion.

3) Negative selection: The biological immune system recognizes all cells within the body and classifies them as self and non-self. Non-self cells are further classified so that the appropriate defence mechanism can be initiated. Negative selection provides tolerance to self cells. During the generation of T-cells - components of cell-mediated immunity- receptors are made through a pseudo-random genetic rearrangement. Then they undergo negative selection in which T-cells that react with self-proteins are destroyed. These matured cells circulate throughout the body to perform immunological functions. The negative selection algorithm has attracted a lot of research attention and undergone significant evolution since it was proposed. There exist several negative selection algorithms that differ from each other in terms of data and detector representation, generation/elimination mechanism and matching rule. However, the two important aspects of all these algorithms are:

1) The target concept is the complement of a self test.
2) The goal is to discriminate between self and non-self patterns.

To implement a basic AIS, four decisions have to be made: encoding, similarity measure, selection and mutation. Once an encoding has been fixed and a suitable similarity measure is chosen, the algorithm then performs selection and mutation, based on the similarity measure, until the stopping criteria are met. Typically, an antigen is the target or solution, e.g. the data item which needs to be checked if it is an intrusion. The antibodies are the remainder of the data. Antigens and antibodies are represented or encoded in the same way. For most problems, the most obvious representation is a string of numbers or features, where the length is the number of variables, the position is the variable identifier and the value is the binary or real value of the variable. The similarity measure or matching rule is the most important design choices in developing an AIS algorithm, and is closely coupled to the encoding scheme. If the encoding is binary, Hamming distance can be a good measure of similarity. If the representation is non-binary, there are even more possibilities to compute the distance between the two strings (e.g., Euclidian distance). The most commonly used mutation operators in AISs are similar to those in GA. For binary strings, bits are flipped; for real value strings, an element is changed at random; and for other types, the elements are swapped.

In addition to the algorithms mentioned above, there exist immune algorithms like immune genetic algorithm, agent algorithm and immune algorithm with bacteria. Algorithms of AIS have found successful practical application in computer security, fault detection, abnormality detection, optimization and data mining.

F. Reinforcement Learning

Conventional artificial intelligence is based on machine learning, which is the development of the techniques and algorithms that allow machines to simulate learning. Machine learning attempts to use computer programs to generate patterns or rules from large data sets. RL [41], [42] is a sub-area of machine learning concerned with how an agent should take actions in an environment so as to maximize some notion of a long-term reward.

RL is biologically inspired and acquires its knowledge by actively exploring its environment. At each step, it selects some possible action and receives a reward from the environment for this specific action. Note that the *best* possible action at some state is never known a-priori. Consequently, the agent has to try many different actions and sequences of actions and learns from its experiences.

Usually, reinforcement learning tasks are described as a Markov Decision Process (see Figure 5), consisting of an agent, set of possible states S, set of possible actions $A(s_t)$ for all possible states s_t and a reward function $R(s_t, a_t)$, specifying the environment reward to the agent's selected action. Additionally, the *policy* π_t defines how the learning agent behaves at some time-step t. The optimal policy is usually defined as $\pi*$. The *value function* $V(s_t, a_t)$ defines the expected total reward when taking action a_t in state s_t, if from the next state s_{t+1} the optimal policy $\pi*$ is followed. This is the function the agent has to learn in order to achieve the optimal policy.

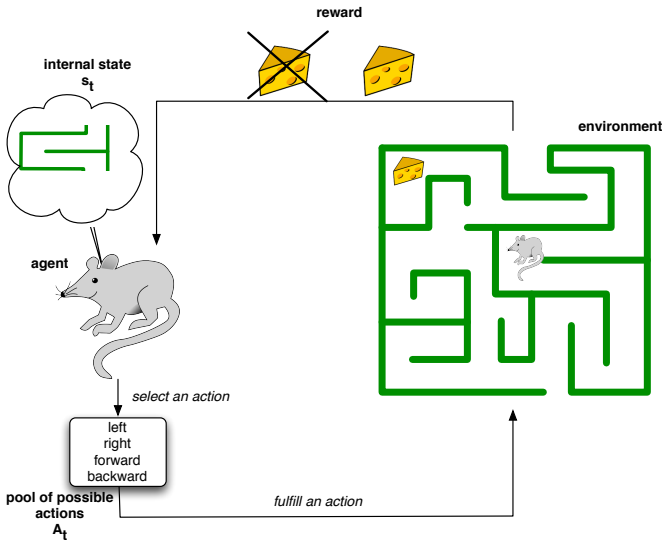

Fig. 5. Reinforcement learning model

RL is well suited for distributed problems, like routing. It has medium requirements for memory and computation at the individual nodes. This arises from the need of keeping many different possible actions and their values. It needs some time to converge, but is easy to implement, highly flexible to topology changes and achieves optimal results.

Q-learning: One simple and powerful RL algorithm is Q-learning [43]. It does not need any model of the environment and can be used for online learning of the value function of some RL task, referred to as the Q-value function. Q-values are updated as follows:

$$Q(s_{t+1}, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \phi \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (6)$$

Where $Q(s_t, a_t)$ is the current value of state s_t, when action a_t is selected. The algorithm works as follows: at some state s_t, the agent selects an action a_t. It finds the maximum possible Q-value in the next state s_{t+1}, given that a_t is taken and updates the current Q-value. The discounting factor $0 < \phi < 1$ gives preference either to immediate rewards (if $\phi \ll 1$) or to rewards in the future (if $\phi \gg 0$). The learning constant $0 < \alpha < 1$ is used to tune the speed of learning to achieve better convergence and stability of the algorithm.

Q-learning has been already widely applied to the wireless ad hoc scenario, for example for routing in WSNs [44]–[47]. It is very easy to implement and has a good balance of optimality to memory and energy requirements.

Dual RL: Dual RL is very similar to Q-learning. However, the reward function uses the best Q-values of the next state *and* the previous one. This increases slightly the communication overhead, but speeds up learning. The DRQ-Routing [48] protocol is based on it and optimizes point-to-point routing. However, compared to approaches with Q-learning, the implementation is more complex.

TPOT Reinforcement Learning: Team-partitioned, opaque-transition reinforcement learning (TPOT-RL) has been developed for simulated robotic soccer [49]. It allows a team of independent learning agents to collaboratively learn a shared task, like soccer playing. It differs from traditional RL in its value function, which is partitioned among the agents and each agent learns only the part of it directly relevant to its localized actions. Also, the environment is *opaque* to the agents, which means that they have no information about the next possible actions of their mates or their goodness.

TPOT-RL is fully distributed and achieves good results. However, its implementation is not trivial and it requires additional communication cost, since rewards are sent to the agents only after concluding the task (in the network context: after delivering the packet at the destination). It has been implemented for routing in TPOT-RL-Routing [50].

Collaborative RL: A formal definition of RL in a distributed environment and a learning algorithm is given in [51]. It presents a reinforcement learning algorithm, designed especially for solving the point-to-point routing problem in MANETs. Collaborative RL is greatly based on Q-learning, but uses also a decay function (very similar to pheromone evaporation in ACO, see Section III-D) to better meet the properties of ad hoc networks. The approach is feasible also for routing in WSNs, since it requires only minimal cost overhead.

An overall high-level comparison of CI approaches in terms of computational requirements, memory requirements, flexibility and optimality of results is given in Table I. Solutions that require a heavy computational or storage cost are suitable for centralized computations carried out by a base station. On the other hand, solutions that require less computations and storage are suitable for distributed computations carried out at each node. Besides, some paradigms assure optimal solutions, and some give sub-optimal solutions. Optimality is measured as the ability of the CI paradigm to find the best possible solution to some problem and not just an approximation. For example, when using RL for finding shortest paths for routing, it is indeed able to find them. Flexibility refers to the ability of the algorithms to adapt to a changing environment, like failing nodes. For superior performance, the designer has to properly choose a paradigm or a combination based on the exact application scenario and requirements.

IV. Design, Deployment, Localization, and Security

The CI algorithms discussed in this paper have been tailored or hybridized to suit the challenges in WSNs. The subsequent sections have been organized into three different classes of WSN problems: This section starts with design, deployment, localization and security issues. The next Section V presents a discussion on energy-aware routing and clustering for WSNs and the survey is completed with an overview of scheduling, data aggregation, sensor fusion and QoS management problems and their CI-based solutions in Section VI. Each subsection in this, and the following two sections begins with a brief summary on suitability of the CI approaches for the challenge under discussion and pros and

TABLE I
PROPERTIES OF BASIC CI PARADIGMS

CI Approach	Computational requirements	Memory requirements	Flexibility	Optimality
Neural networks	Medium	Medium	Low	Optimal
Fuzzy Logic	Medium	Medium	High	Optimal
Evolutionary algorithms	Medium	High	Low	Optimal
Swarm intelligence	Low	Medium	High	Optimal
Artificial Immune Systems	Medium	Problem dependant	High	Near Optimal
Reinforcement learning	Low	Medium	High	Optimal

cons of approaching the challenge with CI techniques. The following Section VII summarizes the findings in the survey and makes suggestions on which CI technique to focus on when solving a particular problem.

A. Design and Deployment

CI techniques can be very helpful in the process of designing and planning the deployments of sensor networks and there have been many efforts to apply them in this context. Table II lists some CI applications in WSN design and deployment; and the applications are discussed in the following paragraphs.

Deployment of the nodes of a WSN can be planned or ad hoc depending on applications. Node positions of a WSN for industrial or health monitoring application are determined beforehand. On the other hand, nodes of a WSN for disaster monitoring are deployed in an ad hoc manner. Coverage and connectivity are among the important considerations in making deployment decisions.

1) Fuzzy Logic: Fuzzy logic has been proposed for deployment in [52]. This techniques assumes that the area to be monitored by a sensor network is divided into a square grid of sub-area, each having its own terrain profile and a level of required surveillance (therefore, its own path loss model and required path loss threshold). The proposed technique uses fuzzy logic to determine the number of sensors $n(i)$ necessary to be scattered in a subarea i. For a sub-area i, path loss $PL(i)$ and threshold path loss PL_{TH} are normalized on a scale 0 to 10, then divided into overlapping membership functions low, $medium$ and $high$. This requires $3^2 = 9$ rules. The system computes an output $weight(i)$, from which the number of sensors is determined as

$$n(i) = \frac{weight(i)}{\sum_j weight(j)} \tag{7}$$

The article shows that the fuzzy deployment achieves a significant improvement in terms of worst case coverage in comparison to uniform deployment.

2) Evolutionary Algorithms: A decision support system based on coarse grain decomposition GA is proposed in [53]. The decision support system is meant for the use of a process engineer who interacts with it to determine optimal sensor network design as outlined in Figure 6.

The process engineer introduces information about the process under stationary operating conditions through a rigorous module called model generator, which involves linear

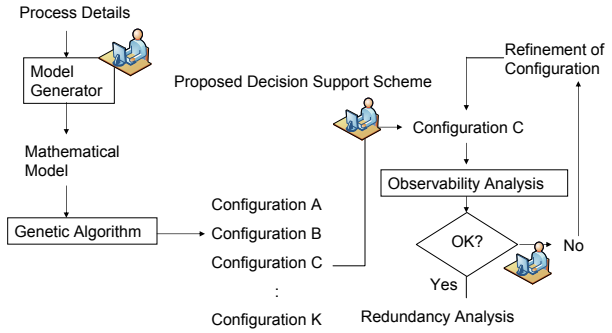

Fig. 6. The role of GA in decision support system for sensor network design

functionalities and non-linear equations. Then, the engineer chooses among the different candidates the initial sensor network. The chosen configuration undergoes observability analysis which determines which of the unmeasured variables are observable. Based on the results, the engineer decides whether the information achieved from the configuration is satisfactory or not. If not, the sensor network needs to be improved by adding more instruments before the analysis is repeated. Traditionally, the analysis begins from an initial sensor network with a few instruments chosen by the process engineer based on his skills and experience. Then, an iterative process takes place; the process engineer runs the tools in the decision support system and refines the configuration until satisfactory instrumentation is achieved. In the case of complex industrial processes, a complete iteration involves a great deal of expert examination and several executions of the analysis software tools. Therefore, a good initial configuration assures lesser number of iterations before the final configuration is reached. The authors propose the use of a GA to determine various initial configurations.

Each chromosome used in GA is a sequence of length l which represents number of variables in the mathematical model of the network, like in 10001100. Here, a 1 represents the presence of a sensor to measure the variable at that position in the mathematical model, and l is the number of variables in the mathematical model. GA uses binary tournament to select individuals to go to next generation maintaining the best up-to-the-moment individual with the elitist approach. The entire set of parents is replaced by offspring. One-point crossover and bit-flip mutation operators are used to improve exploitation and exploration respectively. The fitness function,

TABLE II
A SUMMARY OF CI APPLICATIONS IN WSN DESIGN AND DEPLOYMENT SURVEYED IN SUBSECTION IV-A

CI Paradigm	Algorithm	Articles	Simulation/Real-deployment/Overview	Centralized/Distributed
Fuzzy logic	Fuzzy-Deployment	[52]	Simulation	Centralized
EA	Coarse Grain Decomposition GA	[53]	Real deployment	Centralized
SI	Sequential PSO	[54]	Simulation	Centralized
	PSO-SinkPath*	[55]	Simulation	Centralized
	Traffic-PSO*	[56]	Real deployment	Centralized
RL	COORD	[57]	Simulation	Distributed
	Service Directory Placement Protocol	[58]	Simulation	Semi-distributed

*This algorithm is renamed here for easy reference.

which GA seeks to maximize, is defined as

$$f(i) = N_R(i) + N_{obs}(i) + 1 - N_C(i) \tag{8}$$

where $N_R(i)$, N_{obs} and $N_C(i)$ are the normalized values corresponding to the reliability, observability and cost terms, respectively. The proposed scheme is tested on a ammonia synthesis plant, where all the compositions and critical variables were set as measured. The Performance of an initial sensor configuration suggested by the GA reported to be more cost effective and more reliable than the initial configuration determined by the process engineer. The initial configuration suggested by GA is reported to have achieved a cost saving of 60%, and a smaller amount of time to complete the whole observability analysis.

3) Swarm Intelligence (PSO):

Sequential PSO: A sequential form of PSO is presented for distributed placement of sensors for maritime surveillance application in [54]. Given fixed-size sets S_{T_x} of N_{T_x} transmitters and S_{R_x} of N_{R_x} receivers, the goal is to determine the optimal placement of sonar sensors so that detection coverage is maximized in a fixed volume V representing a maritime region. The sequential PSO algorithm consists of selecting a sensor and subsequently optimizing its placement with respect to the other sensors (assumed fixed) using the standard PSO algorithm. The article shows that this technique can reduce the function evaluations by about 59% and calls to acoustic model procedure by about 92% and achieve about 6% better coverage compared to the standard PSO.

PSO-SinkPath: Yet another application of PSO is detailed in [55]. The goal of this study is to determine the optimal path for sink node movement across the sensor field. The article defines node throughput as the average number of data units forwarded by the sensor node in a time slot; and aggregated throughput is the overall throughput of the sensor network at a given sink node location. Further, average throughput is the average of the aggregated throughput obtained in each point along the selected path. Fitness of a particle is defined as $F = (ratio \times \text{average throughput})$, where $ratio$ is the number of sensor nodes within the communication range of sink node. The results indicate that the approach is good for sparse deployments and degrades as the number of sensors increases. However, good network coverage is achieved.

Traffic-PSO : A topological planning method based on PSO for traffic surveillance application is discussed in [56]. The study described in the article uses a large number of camera loaded sensor nodes, densely deployed over a maze of roads. Some of the nodes need to have higher transmission power. The problem here is to determine optimal allocation of high power transmitters to existing nodes such that maximum coverage is achieved at minimum cost. The concept of small world phenomenon [59] is effectively used in the study. The proposed PSO algorithm is reported to have resulted in symmetric distribution of high power transmitters, improved network performance and enormous saving in system cost.

4) Reinforcement learning: COORD, a distributed reinforcement learning based solution to achieving best coverage in a WSN is presented in [57]. The goal of the algorithm is to cooperatively find the combination of active and sleeping sensor nodes in a sensor network, which is still able to perform full covered sensing of the desired phenomena. For this the authors propose three similar approaches, all based on Q-Learning.

In COORD, nodes are individual learning agents. The actions taken are two: transferring from sleeping to active mode and back. The sensor network is divided into a rectangular grid and the goal is to cover each grid vertex by some sensors, best by exactly one. A Q-value is assigned to each grid vertex, which represents the number of sensor nodes, currently covering this vertex. The probability of choosing an action a in some step is given by the Boltzmann exploration formula:

$$P(a) = \frac{e^{Q(s,t)/T}}{\sum a' e^{Q(s,a')/T}} \tag{9}$$

During the search for an action, the temperature T is decreased in order to increase the probability of selecting the best actions. The Q-value is updated according to the standard update formula (see Section III-F). In each run of the algorithm, each node evaluates its current Q-value table with all grid vertices it covers and takes an action. After that, all nodes evaluate again their Q tables and so on.

The other two solutions are very similar and the results they show are also comparable. The authors claim that COORD performs best, but the presented measurements show clearly that all three learning based approaches perform very similar. A comparison to some state-of-the-art approach is not provided and thus the results cannot be properly evaluated. Also, a clear protocol implementation is missing, leaving open many questions about coordination and exchange of Q-values and the states of the grid vertices. However, the approach is fully distributed and can be run online if needed. Also, it shows a nice modeling work of converting a centralized problem into a distributed one and solving it with RL.

SSDP: The study reported in [58] presents a reinforcement learning based approach for service positioning in MANET. The system is presented as a semi-Markov decision process and the optimal behavior is learned via Q-learning. The learning agent is situated together with the service provider on one of the hosts in the network and has the ability to move to other hosts. Thus, only one learning agent is present in the system (with more service providers more agents have to be deployed). The system state is given through different query-related parameters, like queries' average hop count, number of neighboring clients, etc. The protocol is called service directory placement protocol and it can be run together with any other service discovery protocols.

B. Localization

GA and PSO based localization algorithms have been proposed recently, some of which are discussed here. The algorithms are listed in Table III.

Most WSN localization algorithms share a common feature that they estimate the location of nodes using the a priori knowledge of the positions of special nodes called beacon nodes or anchor nodes. The most intuitive solution to the localization problem is to load each node with a global positioning system. But it is not an attractive solution because of cost, size and power constraints. Typically, the nodes estimate their locations using signal propagation time or received signal strength. Signal propagation time is estimated through measurement of time of arrival, round trip time of flight or time difference of arrival of two signals. The localization is formulated as a multidimensional optimization problem, and tackled with population based CI methods such as GA and PSO.

1) Genetic Algorithm: A GA based node localization algorithm *GA-Loc* is presented in [60]. Each of the N non-anchor nodes in the study is assumed to have ability to measure its distance form all its one-hop neighbors. GA estimates the location of node i (x_i, y_i) by minimizing the objective function OF defined in (10),

$$OF = \sum_{i=1}^{N} \sum_{j \in N_i} \left(\hat{d}_{ij} - d_{ij} \right)^2 \tag{10}$$

where N_i is a set of neighbors of node i, $\hat{d}_{ij}$ is the estimate of the distance between node i and its neighbor node j and d_{ij} is the measured distance between node i and its neighbor node j.

The real number coded GA uses roulette wheel selection and three types of crossover operators order crossover, partially mapped crossover and cycle crossover. The authors present results of MATLAB simulations of different scenarios that involve 100 nodes, eight or ten anchors, and 5% or 10% Gaussian noise in distance measurement. The results show very small (less than 1%) localization error in all simulations. The algorithm seems to produce accurate localization, but involves an overhead: A central node needs to collect from all nodes their measurement of distances from all neighbors. In addition, localization results produced by GA need to be propagated back to all nodes. The algorithm has a scalability limited by computational resources in the node which runs the GA.

Two-phase localization: A two-phase centralized localization scheme that uses simulated annealing and GA is presented in [61]. The study approaches the problem in a slightly different way: it attempts to minimize a performance measure J which takes a node's distance from all its neighbors.

$$\min_{\hat{x},\hat{y}} \left\{ J = \sum_{i=M+1}^{N} \sum_{j \in N} (\hat{d}_{ij} - d_{ij})^2 \right\}, d_{ij} \leq R, j \in N_i \tag{11}$$

Here $d_{ij} = \sqrt{(\hat{x}_i - \hat{x}_j) + (\hat{y}_i - \hat{y}_j)^2}$, $(\hat{x}_i, \hat{y}_i)$ are estimated coordinates of node i, $(\hat{x}_j, \hat{y}_j)$ are estimated coordinates of one hop neighbor j of node i. N_i is a set of neighbors of node i and R is the radio range. The study proposes two variants of the two-phase localization method. In the first phase, the initial localization is produced. The result of the first phase is modified in the second phase. Authors have used two methods, simulated annealing and GA, for the second phase. In the first phase, only the nodes with three anchor neighbors are localized. All nodes are divided into two groups: group A containing M nodes with known location (in the beginning, only the anchor nodes) and group B of nodes with unknown location. In each step, node i, $i = M+1, 2, \ldots, N$ from the group B is chosen. Then, the three nodes from the group A that are within the radio range of node i are randomly selected. If such nodes exist, the location of node i is calculated based on true inter-nodes distances between three nodes selected from group A and the measured distances between node i and these three nodes. The localized node i is moved to the group A. Otherwise, another node from the group B is selected and the operation is repeated. The algorithm is terminated when there are no more nodes that can be localized based on the available information.

The initial generation of GA used in the second phase has a set of nodes whose coordinates are determined in the first phase. The GA uses tournament selection, discrete recombination crossover and random mutation. Simulation

TABLE III
A SUMMARY OF CI APPLICATIONS IN WSN LOCALIZATION SURVEYED IN SUBSECTION IV-B

CI Paradigm	Algorithm	Articles	Simulation/Real-deployment/Overview	Centralized/Distributed
EA	GA-Loc*	[60]	Simulation	Centralized
	Two-Phase GA-Loc*	[61]	Simulation	Centralized
	Two-Phase Localization Algorithm	[62]	Simulation	Centralized
PSO	PSO-Loc*	[63]	MATLAB simulation	Centralized
PSO	Iterative PSO	[64]	MATLAB simulation	Distributed

*This algorithm is renamed here for easy reference

results are presented. Besides, the results are compared with those obtained using semidefinite programming [65]. Both GA and simulated annealing have shown achieve less than 1% localization error. However, simulated annealing is shown to perform better than GA both in terms of localization error and computation time.

Two-Phase Localization Algorithm: Another two-phase centralized localization method that uses a combination of GA and simulated annealing algorithm proposed in [62] addresses a problem called flip ambiguity problem. It is likely that in distance-based localization, a node is not uniquely localized if its neighbors are on collinear locations. In such topology, the node can be reflected across the line that joins its neighbors while satisfying the distance constraint. The paper proposes a two-phase localization algorithm in which in the first phase, GA obtains an accurate localization estimation. In the second phase, simulated annealing algorithm refines the location estimation of only those nodes that have been wrongly localized due to flip ambiguity. The information on the number of nodes affected by flip ambiguity is collected from the neighborhood information.

The GA in the first phase uses the objective function defined in (11). Chromosomes are coded as sequences of real numbers that represent x and y coordinates of the nodes to be localized. Authors define a *single vertex neighborhood mutation* operator and a *descend based arithmetic crossover* operator. If a node i that does belong to neighborhood of node j in its localization, but still has its transmission radius R less than the distance d_{ij} is treated as wrongly localized. Simulated annealing algorithm used in the second phase attempts to minimize the objective function defined in (12), where l_{ij} represents the measured distance between nodes i and j.

$$\min_{x_i,y_i} \sum_{i=m+1}^{n} \left(\sum_{j \in N_i} (d_{ij} - l_{ij})^2 + \sum_{\substack{j \notin N_j \\ d_{ij} < R}} (d_{ij} - R)^2 \right) \quad (12)$$

The algorithm is reported to have outperformed the semi-definite programming algorithm in all presented simulations.

2) SI (PSO): A PSO based localization scheme *PSO-Loc* is presented in [63]. The objective of the scheme is to estimate x and y coordinates of n nodes in a network of m nodes deployed in 2-dimensional plane. The remaining $M = (m-n)$ nodes are anchor nodes. The localization problem is treated as a $2n$-dimensional optimization problem. The base station runs PSO to minimize the objective function defined in (13).

$$f(x,y) = \frac{1}{M} \sum_{i=1}^{M} \left(\sqrt{(x-x_i)^2 + (y-y_i)^2} - \hat{d}_i \right)^2 \quad (13)$$

where (x, y) is the node location that needs to be determined, $M \geq 3$ is the number of anchors and (x_i, y_i) is the location of anchor i. Here $\hat{d}_i$ is the measured value of distance d_i between a node and the anchor i, obtained under noise. The authors simulate noise as $\hat{d}_i = d_i + n_i$, where n_i is a zero mean additive Gaussian variable with variance σ_d^2. In addition to σ_d^2, the density of anchor nodes affects the accuracy of localization. The paper presents MATLAB™ simulation results in terms of localization error versus various settings of anchor node density and noise variance. The paper compares the performance of PSO with that of a simulated annealing [66], and shows that localization error in PSO is more than halved in all experimental setups.

An extension of this work is reported in [64], which uses PSO and bacterial foraging algorithm[1] for range-based distributed iterative node localization. In this approach, each node runs two-dimensional PSO (or bacterial foraging algorithm) to estimate its own x and y coordinates in a plane mission space. The nodes that get localized with the help from beacons act as beacons themselves in the next iteration. This approach can mitigate inaccuracies due to flip ambiguity in the sense that a node that got localized with the help of three near collinear beacons in an iteration may get more reference nodes in the next iteration. The study makes a comparison between PSO and bacterial foraging algorithms as optimization tools. PSO is reported to be faster than bacterial foraging algorithm. However, the bacterial foraging algorithm is reported to be less memory intensive and more accurate.

C. Security

Surprisingly, WSN security does not seem to be a fertile area for CI applications compared with other issues. The literature has a couple of articles that use CI approaches for WSN security, one of which is discussed below. The applications available in the literature have tackled the issue of denial of service attacks at the node level. It is obvious that the nodes are resource constrained, and therefore, algorithms that involve high computational and storage burdens are not attractive. Fuzzy logic and compact NNs are among solutions discussed here, as summarized in Table IV.

Neural Networks: Many types of DoS attacks on WSNs have been devised. In collision attacks, attackers transmit packets regardless of status of the medium. These packets collide with data or control packets from the legitimate sensors. In unfairness attacks, adversaries transmit as many packets as possible after sensing that the medium is free. This prevents the legitimate sensors from transmitting their own packets. In exhaustion attacks, adversaries transmit abnormally large number of *ready-to-send* packets to normal sensor nodes, thereby exhausting their energy quickly.

Multilayered perceptron and generalized neuron-based distributed secure MAC protocols are proposed in [68], [69] respectively. In these approaches, NNs onboard WSN nodes monitor traffic variations, and compute a suspicion that an attack is underway. When the suspicion grows beyond a preset threshold, the WSN node is switched off temporarily. A detects an attack by monitoring abnormally large variations in sensitive parameters: collision rate R_c (number of collisions observed by a node per second), average waiting time T_w (waiting time of a packet in MAC buffer before transmission), and arrival rate (R_{RTS}), rate of RTS packets received by

[1]Bacterial foraging algorithm is a new population-based multidimensional optimization approach that mimics foraging behavior of *E. coli* bacteria that live in human stomach [67].

TABLE IV
A SUMMARY OF CI APPLICATIONS IN WSN SECURITY SURVEYED IN SUBSECTION IV-C

CI Paradigm	Algorithm	Articles	Simulation/Real-deployment/Overview	Centralized/Distributed
NN	MLPMAC*	[68]	PROWLER Simulation	Distributed
	GNMAC*	[69]	PROWLER Simulation	Distributed
Fuzzy logic	FS-MAC	[70]	OPNET Simulation	Distributed

*This algorithm is renamed here for easy reference

a node successfully per second. The strategy used in both the articles is the same, but the nonlinear mapping between traffic pattern variation and suspicion factor is implemented by different types of NNs in these articles. MLP and generalized neuron are trained using PSO.

These distributed methods ensure that only the part of the WSN under attack is shut down, which is an advantage. Besides, these methods have been reported to be quite effective against DoS attacks, but the effectiveness depends on the value of the threshold suspicion. If the traffic variation under normalcy is unpredictable, then these methods are likely to generate false alarms, which may shut down a node when an important event occurs in the environment.

Fuzzy Logic: In the study presented in [70], the probability that a denial of service attack has been perpetrated is estimated using a decision function that involves two parameters, which are determined using the steepest gradient descent algorithm. A fuzzy logic approach towards secure media access control is presented by the same authors in [71] for enhanced immunity to collision, unfairness and exhaustion attacks. The study aims at detecting and counteracting DoS attacks from adversaries through innovative use of fuzzy logic on each node in the network.

In the proposed algorithm, values that represent R_c and R_{RTS} are categorized into 2 levels, $high$ and low, and values that represent T_w are categorized into 2 levels, $long$ and $short$. The result, the possibility that an attack is detected, is divided into 5 levels: $very\ low$, low, $moderate$, $high$ and $very\ high$. From 3 variables that in 2 classes, 2^3 (=8) rules are formed. The study uses OPNET to simulate a network of 30 sensor nodes having a constant energy deployed randomly in an area of 1 km $\times$ 1 km. Each attack is launched alone for a random duration. Performance of the proposed protocol is compared with that of CSMA/CA. The results show that the former protocol offers a 25% increase in successful data packet transmission, 5% less energy consumption per packet. In each type of attack, the proposed protocol extends first node death time in the network by over 100% as compared to CSMA/CA.

V. ENERGY AWARE ROUTING AND CLUSTERING

A list of the CI based approaches for energy aware routing presented here is given in Table V. A survey and guide of RL routing approaches for telecommunication networks was performed in [72] and a networking perspective of machine learning for wireless sensor networks is presented in [73].

Different approaches have been taken by different researchers for energy aware routing. All-in-one distributed real-time algorithms have proved to work well under WSN-specific requirements like communication failures, changing topologies and mobility. Such approaches are usually based on RL, SI or NNs. However, these algorithms have different properties, which need to be considered depending on the application and deployment scenario. For example, ant-based routing is a very flexible algorithm, but generates a lot of additional traffic because of the forward and backward ants that move through the network. In a WSN scenario, this overhead has to be carefully considered and compared to that in other protocols, which may not provide a very flexible routing scheme, but a lower overhead. RL algorithms have proved to work very well for routing and can be implemented at nearly no additional costs. They should be the first choice when looking for a flexible and low-cost routing paradigm.

On the other hand, algorithms which require an offline learning phase, like GAs or offline NNs, can neither cope up with changing properties of the network, nor provide an energy efficient routing scheme. They require very high costs of gathering first the relevant data on a base station, then calculating the routing tree and then disseminating the tree roles to the nodes. Communication failures cannot be considered and in case of a topology change the whole procedure has to be started from the beginning.

1) Neural Networks: A QoS driven routing algorithm called sensor intelligence routing, is presented in [74] for automatic reading of public utility meters. The study utilizes an NN in every node to manage the routes for data flow. A modification over Dijkstra algorithm proposed in the article finds minimum cost paths from base station to every node. The edge cost parameter between a pair of nodes w_{ij} is determined in each node by a self organizing map, based on QoS parameters latency, error rate, duty cycle and throughput. Each node pings each of its neighbors to find out the quality of link. First layer of SOM has 4 neurons and the second layer has 12 neurons in a 3×4 matrix. Inputs to the SOM are latency, throughput, error-rate, and duty-cycle. The samples presented as inputs to the SOM form groups in such a way that all the samples in a group have similar characteristics. This gives a map formed by clusters, where every cluster corresponds with a specific QoS and is assigned a neuron of the output layer. After a node has collected a set of input samples, it runs the wining neuron election algorithm. After the winning neuron is elected, the node uses the output function to assign a QoS estimation. QoS is obtained in the samples allocated in the cluster, a value between 0 and 10 is assumed. The highest assignment corresponds to scenario in which the link measured has the worst QoS predicted. Lowest assignment corresponds to scenario in which the link measured has the best QoS predicted. Performance of SIR is compared with that of *energy aware routing* and *directed diffusion*. Two case studies are discussed, one with an assumption of all healthy nodes, and the other with 20% failed nodes. In both cases, average latency and

TABLE V
A SUMMARY OF CI APPLICATIONS FOR ENERGY AWARE ROUTING AND CLUSTERING IN WSNS SURVEYED IN SUBSECTION V

CI Paradigm	Algorithm	Articles	Simulation/Real-deployment/ Overview	Centralized/ Distributed
NN	Sensor Intelligence Routing	[74]	OLIMPO simulation	Distributed
Fuzzy logic	Fuzzy-Clusters*	[75]	Java simulation	Centralized
EA	GA-Routing*	[76]	Java simulation	Centralized
	TwoTier-GA*	[77]	Java simulation	Centralized
	Energy Efficient GA Clustering *	[78]	Simulation	Centralized
	Multi-objective DE	[79]	Simulation	Centralized
SI	PSO with time varying inertia weight	[80]	Simulation	Centralized
	PSO with time varying acceleration constants	[80]	Simulation	Centralized
	hierarchical PSO with time varying acceleration constants	[80]	Simulation	Centralized
	PSO with supervisor student mode	[80]	Simulation	Centralized
	Ant-based	[72], [81]–[83]	Overview	
	Collaborative Clustering	[84]	Prowler simulation	Distributed
	Ant-Building*	[85]	Simulation	Distributed
RL	FROMS	[46]	OMNeT++ simulation, testbed with MSB430 sensor nodes	Distributed
	Clique	[86]	OMNeT++ simulation	Distributed
	Q-Routing	[45]	Simulation	Distributed
	DRQ-Routing	[48]	Simulation	Distributed
	Q-RC	[44]	Simulation	Distributed
	RL-Flooding*	[87]	Prowler simulation	Distributed
	TPOT-RL	[50]	Simulation	Distributed
	SAMPLE	[51]	NS-2 simulation	Distributed
	AdaR	[88]	Simulation	Centralized
	Q-PR	[47]	Simulation	Distributed
	Q-Fusion	[89]	Simulation	Distributed
	RLGR	[90]	NS-2 simulation	Distributed

*This algorithm is renamed here for easy reference.

power dissipated are studied. SIR shows superior performance in all cases, especially so when the percentage of dead nodes is high. However, the approach is expensive in terms of the pings to neighboring nodes in order to learn the quality of links. Besides, implementation of a SOM on a node entails computational expenses.

2) Fuzzy Logic: Judicious cluster head election can reduce the energy consumption and extend the lifetime of the network. A fuzzy logic approach based on energy, concentration and centrality is proposed for cluster head election in [75]. The study uses a network model in which all sensor nodes transmit the information about their location and available energy to the base station. The base station takes into account the energy each node has, the number of nodes in the vicinity and a node's distance from other nodes into account and determines which nodes should work as the cluster heads. The base station fuzzifies the variables *node energy* and *node concentration* into three levels: low, $medium$ and $high$, and the variable *node centrality* into $close$, $adequate$ and far. Therefore, 3^3 (=27) rules are used for the fuzzy rule base. The fuzzy outcome that represents the probability of a node being chosen as a cluster head, is divided into seven levels: $very\ small$, $small$, $rather\ small$, $medium$, $rather\ large$, $large$, and $very\ large$. Triangular membership functions are used to represent the fuzzy sets $medium$ and $adequate$ and trapezoidal membership functions to represent low, $high$, $close$ and far fuzzy sets. The article observes a substantial increase in the network life in comparison to the network that uses the low energy adaptive clustering hierarchy approach. For a 20-node scenario in a 100m $\times$ 100m field, the number of data rounds before first-node-death in case of the proposed method is on average about 1.8 times greater than in low energy aware clustering hierarchy. Once again, the approach involves the overhead of collecting necessary information at a base station before determining cluster heads.

3) Evolutionary Algorithms: A GA based multi-hop routing technique named *GA-Routing* is proposed in [91] for maximizing network longevity in terms of time to first node death. The challenge here is to come up with a routing algorithm that maximizes the number of rounds before the first node dies. The GA approach proposed here generates aggregation trees, which span all the sensor nodes. Although the best aggregation tree is the most efficient path in the network, continuous use of this path would lead to failure of a few nodes earlier than others. The goal of this study is to find an aggregation tree, and the number of times a particular tree is used before the next tree comes in force.

A gene represents an s bit number that denotes the frequency and a chromosome contains k genes, each representing a spanning tree Chromosome length is $s \times k$ bits.

Simulation results show that GA gives better lifetime than the *single best tree* algorithm, and GA gives the same lifetime as the cluster based maximum lifetime data aggregation algorithm [92] for small network sizes. However, the algorithm is centralized and the cost of disseminating the optimal routing paths to the nodes is not considered.

Two-Tier GA: Another application of GA for maximum lifetime routing for a two-tier network is presented in [77]. In a two tier sensor network, a few sensor nodes having higher power levels act as cluster heads. If a cluster head fails due to loss of power, the whole cluster ceases to operate. This influences load on the existing nodes, causing them to deplete

their power quickly. Flow-splitting type of routing algorithms are used in such eventuality. The approach discussed in the article uses GA determine a non-flow-splitting routing strategy.

A chromosome is represented as a string of node numbers for a particular routing scheme. Length of chromosome is equal to number of relay nodes. Selection of individuals is carried out by using Roulette-Wheel selection method. Uniform crossover is used with swapping probability of 0.5. The node in the chromosome, which dissipates maximum energy is selected as critical node. Destination of critical node is changed to a randomly selected node. Average life time extension of 200% is observed in comparison to the networks that use minimum transmission energy model and minimum hop routing model discussed in [93] and [94] respectively.

Energy Efficient GA Clustering: Another application of GA in energy efficient clustering is described in [78]. The proposed GA represents the sensor nodes as bits of chromosomes, cluster heads as 1 and ordinary nodes as 0. The number of bits in a chromosome is equal to the number of nodes. Fitness of each chromosome is computed as

$$F = \sum_{i} \alpha(w_i, f_i), \forall f_i \in \{C, D, E, SD, T\} \tag{14}$$

where C represents sums of distances of nodes to cluster heads, and distance from cluster head to the sink;

$$C = \sum_{i=1}^{k} d_{ih} + d_{hs} \tag{15}$$

D represents sums of distances from cluster heads from sink nodes;

$$D = \sum_{i=1}^{m} d_{is} \tag{16}$$

SD represents the standard deviation of cluster distances, where μ is the average of cluster distances; and

$$SD = \sqrt{\frac{1}{h} \sum_{i=1}^{h} (\mu - d_{cluster_i})^2} \tag{17}$$

E represents the energy consumed in transferring aggregated message from the cluster to the sink.

$$E = \sum_{j=1}^{k} E_{T_{jh}} + k \cdot E_R + E_{T_{hs}} \tag{18}$$

Initially, weights are assigned arbitrary weights w_i, and then weights are updated as

$$w_i = w_{i-1} + c_i \cdot \Delta f_i \tag{19}$$

where $\Delta f_i = f_i - f_{i-1}$ and $c_i = \frac{1}{1+e^{-f_{i-1}}}$. The results show that the GA approach possesses better energy efficiency than do hierarchical cluster based routing and LEACH. However, with GA, there is an overhead of gathering critical information about the whole network on a base station, before determining the clusters.

There are also some other similar ideas based on GAs, where a base station computes the optimal routing, aggregation or clustering scheme for a network based on the information about full topology, remaining energy on the nodes etc. Such algorithms are only feasible if the network is expected to have a static topology, perfect communication, symmetric links and constant energy. Under these restrictions, a centrally computed routing or aggregation tree makes sense and is probably easier to implement. However, these properties stay in full conflict with the nature of WSNs.

Multi-objective DE: The study presented in [79] proposes a multi-objective differential evolution algorithm to determine a set of Pareto optimal routes with respect to energy consumption, latency and channel capacity for single and multipath routing problems[2]. Multi-objective differential evolution produces multiple candidate routes that represent different possible tradeoff between energy consumption and latency for a communication in completely connected network in which each node knows precise location of every other node.

A chromosome in multi-objective DE consists of a sequence of network node IDs with the source node in the first locus and the destination node in the last locus. Each parent p_i in the population undergoes a reproduction operation to produce an offspring p_i' according to

$$p_i' = \gamma \cdot p_{best} + (1 - \gamma) \cdot p_i + F \cdot \sum_{k=1}^{K} (p_{i_a^k} - p_{i_b^k}) \tag{20}$$

where p_{best} is the best individual in the parent population, γ represents greediness of the operator, and K is the number of perturbation vectors, F is the scale factor of the perturbation, $p_{i_a^k}$ and $p_{i_b^k}$ are randomly selected distinct individual pairs in the parent population, and p_i' is the offspring that is generated; γ, K, and F are the parameters associated with the algorithm. The whole population is divided into multiple ranks based on the Pareto optimality concept. All parents and their offspring compete for entering into the next generation based on their ranks and the crowd distance metric σ_{crowd} which indicates similarity of solutions within the same rank. In discrete version used in the article, three probabilities: greedy probability p_g, mutation probability p_m, and perturbation probability p_p are introduced.

The article presents simulation results which demonstrate the ability of multi-objective DE to come up with a Pareto front of multiple routes between a given source and destination. At one extreme of the Pareto front exists the route that has minimum latency but maximum energy requirement. At the other, exists the route that requires minimum energy but involves maximum latency. Again, the major weakness of the approach is the necessity of the global view of the network at the base station in order to compute distances between its nodes.

4) Swarm Intelligence: Four variants of PSO, namely PSO with time varying inertia weight, PSO with time varying acceleration constants, hierarchical PSO with time varying acceleration constants and PSO with supervisor student mode, are proposed for energy aware clustering in [80]. In PSO with

[2]In Economics, Pareto optimality is a situation which exists when economic resources and output have been allocated in such a way that no-one can be made better off without sacrificing the well-being of at least one person.

time varying inertia weight, the inertia weight w is decreased linearly from 0.9 in first iteration to 0.4 in the last iteration. In PSO with time varying acceleration constants, inertia weight is set constant, and acceleration constants c_1 and c_2 are varied linearly from 2.5 to 0.5 linearly in every iteration. Therefore, particles move towards the solution in large steps initially, and the step size reduces in every iteration.

In hierarchical PSO with time varying acceleration constants method, the particle update is not influenced by the velocity in previous iteration. Thus, re-initialization of velocity is done when the velocity stagnates in the search space. Therefore, a new set of particles is automatically generated according to the behavior of the particles in the search space, until the convergence criteria is met. The re-initialization velocity is set proportional to $V_{\max}$. Lastly, in PSO student supervisor model, a new parameter called momentum factor (mc) is introduced, and the PSO update equation is modified to (21).

$$x_i^{k+1} = (1 - mc) \cdot x_i^k + mc \cdot v_i^{k+1} \tag{21}$$

PSO assigns n_j nodes to each of the k cluster heads, $j = 1, 2, \cdots, k$ such that the total energy loss due to physical distances E_{dd} is minimum. This is defined in (22), where D_j is the distance between cluster head j and the sink.

$$F = \sum_{j=1}^{k} \sum_{i=1}^{n_j} (d_{ij}^2 + \frac{D_j^2}{n_j}) \tag{22}$$

Clustering is based on a simple idea that for a group of nodes that lie in a neighborhood, the node closest to the base station becomes the cluster head. The approach has a drawback: Clustering depends solely on physical distribution of nodes, but not on energy available to the nodes. The article reports two case studies; in the first, nodes have a fixed transmission range and in the second, nodes do not have any range restrictions. A detailed comparative analysis of the algorithms for optimal clustering is presented in [80]. The results show that PSO with time varying inertia weights performs better in the first case study, and PSO with student supervisor model in the second.

Ant-based approaches: Biologically inspired techniques like SI are popular tools used by researchers to address the issue of energy aware routing.

Introductions to ant inspired algorithms and brief overviews of ant based routing methods in WSNs are given in [82] and [83]. The study presented in [83] investigates ant-based and genetic approaches and the associated techniques for energy aware routing in WSNs. The article presents a mathematical theory of the biological computations in the context of WSNs, and a generalized routing framework in WSNs. An overview of emerging research directions is presented within the biologically computational framework. A survey of adaptive routing methods based on SI is given in [81] and in [72].

Collaborative Clustering Algorithm: In many applications, mere number of living nodes does not represent the effectiveness of the network. For example, a network that has lost a quarter of its nodes from one quadrant is less effective than the network that has lost a quarter of its nodes from the whole area uniformly. Article [84] defines a new parameter for evaluating longevity of wireless sensor networks. The parameter is called *Effectiveness*, and [84] defines it as in (23). The network longevity is defined as the time for which the network *Effectiveness* is equal to or greater than 70%.

$$\textit{Effectiveness} = \sqrt{\frac{\text{Area Covered} \times \text{Surviving Nodes}}{\text{Total Area} \times \text{Total Nodes}}} \tag{23}$$

Biological soldier ants that have the support of other soldier ants are found to be more aggressive in nature. An ant is observed to exhibit higher eagerness to fight when it is amidst strong ants. This fact inspires the collaborative clustering algorithm for wireless sensor network longevity that possesses good scalability and adaptability features. Here, each node has an *Eagerness* value to perform an operation, which is defined as

$$\textit{Eagerness} = BC_{self} + \frac{1}{10} \sum_{i=1}^{N} BC(i) \tag{24}$$

where BC_{self} is the battery capacity of the node, N is the number of neighbors the node has, and $BC(i)$ is the battery capacity of i^{th} neighbor. At a regular periodicity, each node computes its *Eagerness* value and broadcasts over the network. The node that has highest *Eagerness* value decides to act as a cluster head, and the other nodes accept it. The cluster head floods the new clustering information, which helps other nodes to readjust their power levels just enough for them to transmit to the cluster head.

The method assures that only the nodes that have sufficient energy in their reservoir, and have strong neighbors, opt to become cluster heads. The algorithm has overheads due to the fact that each node has to flood its *Eagerness* every now and then. In addition, the traffic of packets might flow away from a sink node just because a node in that direction has higher *Eagerness*. Thus, the algorithm is sub-optimal in terms of minimizing energy of the individual nodes, but optimal in terms of making effective use of the energy available to the whole network.

Results of several case studies conducted on MATLAB-based PROWLER simulator are presented. A case study differs from another in terms of the standard deviation in distribution of the initial energy to the nodes. The network with initial energy varying between 1500mAh to 2400 mAh that uses the proposed algorithm is reported to have over 26% more network life compared to the network without such a clustering scheme. However, the algorithm needs to be evaluated under realistic conditions like link failures and packet loss, which could affect its work significantly.

AntBuilding: Article [85] presents an ant-system based approach for routing in a WSN installed in a building. The authors use software agents as ants in the network, which try out different routes from the source to the destination and, after completing a full tour, return to the source and update the transition probabilities (pheromone levels) on all intermediate nodes. The approach is very similar to AntHocNet [95], and as every other swarm-based one, is fully distributed and is well suited for networks. However, the returning ants in the network create unnecessary overhead for a sensor network.

5) Reinforcement Learning: One of the fundamental and earliest works in packet routing using machine learning techniques is Q-Routing [45]. The authors describe a very simple, Q-learning based algorithm, which learns the best paths considering the least latency to the destinations.

A Q-value is assigned to each pair $(sink, neighbor)$ at all nodes, which is the estimation of how much time the packet will travel to the sink taking this particular next step (including packet queues). Upon sending a packet to some neighbor, a node immediately receives a reward from it, namely the minimum time this neighbor needs to forward the packet to the sink:

$$t = \min_{z \in \text{ neighbors of } y} Q_y(d, z) \tag{25}$$

The new estimate of the remaining time is updated according to:

$$\Delta Q_x(d, y) = \eta(\overbrace{q + s + t}^{\text{new estimate}} - \overbrace{Q_x(d, y)}^{\text{old estimate}}) \tag{26}$$

where η is the learning rate, q is the units of time the packet has to spent in the queue of the current node x, and s is the time for transmission between nodes x and y.

Simulations proved the algorithm to be highly efficient under high network loads and to perform also well under changing network topology. Although the approach was developed for wired, packet-switched networks, it can be easily applied to wireless networks and is especially well suited for WSNs, since it uses only locally available information at the nodes and is fully distributed.

DRQ-Routing: DRQ-Routing [48] is based on Q-Routing and uses the same WSN application scenario: routing packets from a source to a sink, while minimizing delivery time of the packets. However, the authors use dual reinforcement learning (See Section III-F). Thus, learning converges faster and the protocol shows better performance. The approach is again fully distributed and uses only local information and feedback appended to packets from neighboring nodes. However, its communication cost is slightly increased by the backward rewards, compared to Q-Routing.

Q-RC: Q-RC (Q-Routing with Compression) [44] presents a routing protocol, where the goal is to aggregate the source packets as early as possible in the path, compress them and send to a single sink (See Figure 7, taken from [44]).

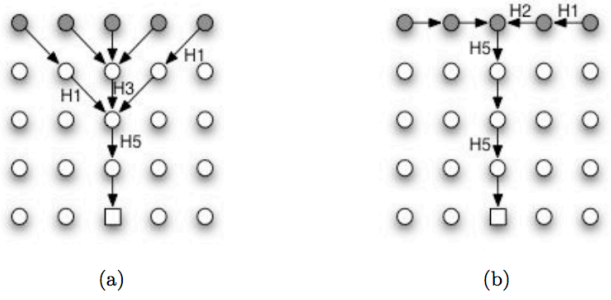

Fig. 7. (a) routing driven compression and (b) compression driven routing

Again, agents are the sensor nodes and the actions are the possible forwarding options to different neighbors. A q-value is associated with each forwarding option and the one with the highest q-value is considered the best. The q-values are initialized according to:

$$Q(a_v) = \beta * Q_1(a_v) + (1 - \beta) * Q_2(a_v) \tag{27}$$

where $Q_1(a_v) \in \{0, 0.5, 1\}$ is the evaluation of making progress to the sink and $Q_2(a_v)$ is the contribution of achieving a good data aggregation. The second part is updated according to incoming rewards, the first part if fixed and pre-known. β is the weighting parameter. Update of the q-values is done according to the standard Q-learning algorithm (See Section III-F). The rewards are calculated with:

$$R(\rho, p, q_{u,best}) = \rho * p + \max_a q_{u,a} \tag{28}$$

where ρ is compression rate achieved from p number of packets at agent u. The exploration strategy is the well known $\epsilon - greedy$.

Results presented include convergence speed and total energy expenditure compared to minimum hop-based routing. The performance is unfortunately not compared to other state-of-the-art protocols and thus it stays unclear how the increased costs of exploration affect the total performance of the system. However, the work is well modeled and presented and the algorithm simple, elegant and very efficient. Beside this, it can be easily adjusted to accommodate other cost metrics and thus completely different system goals.

The best compression path is learned by Q-learning. The approach is fully distributed and can be applied easily to similar routing problems. However, the protocol is somewhat premature, since it gives no communication details, nor an implementation for exchanging the rewards.

An additional improvement of Q-RC is presented in [96], where instead of pre-setting the parameters of the learning algorithm, a Bayesian exploration strategy is used. For this, a new metric is introduced, the value-of-perfect-information and the actions are selected such that the expectation of the q-values *and* and value-of-perfect-information are maximized. To calculate value-of-perfect-information, three values need to be stored at each node: (1) its Manhattan distance from the sink, (2) its residual energy and (3) the number of previous aggregated transmissions. The calculation of the immediate reward changes accordingly. Simulation results show that the cumulated reward during the learning phase of the protocol more than doubles, thus speeding up the convergence.

The paper presents an idea which can be applied to all other RL-based algorithms, which need parameter pre-setting and should be further explored and refined.

AdaR: Yet another Q-Learning based routing mechanism is presented in [88]. It uses an offline learning procedure, based on Q-learning and claims to learn the routing value function faster than traditional Q-Learning. The setting is similar to those presented above: many source nodes are sending data to a single base station. The algorithm takes into account the aggregation ratio, the residual energy on the nodes, the hop

cost to the base station and the link reliability between the nodes.

The algorithm runs in learning episodes. The learning agents are again the nodes and Q-values are assigned to each possible next hop at each node. During each episode, the current Q values are used to route a packet to the base station. At each hop, the full hop information is appended to the packet (residual energy, rewards, etc.). Rewards are generated either at the base station or at intermediate nodes, if some routing threshold is exceeded. When the base station has enough such packets (undefined how many), it calculates the Q values offline for the nodes in the network and disseminates them via a network-wide broadcast.

The approach has some useful properties, like taking into account more than only one routing cost metric. However, it shows also a number of drawbacks. The result section unfortunately tries to evaluate the protocol only in terms of how many such packets are needed to find the end policy compared to a traditional Q-learning protocol. However, in a real world scenario and implementation, the protocol is hard to implement and maintain: a lot of additional communication cost is incurred through the new Q-values broadcasts; the protocol is hardly scalable because of the growing size of the packet at each hop; in case of a link failure or node mobility, a new round of learning episodes has to start to find the new optimal policy. Not less important is the non-distributed manner of learning of the algorithm, which is very different from traditional RL approaches - it needs a whole set of samples (packets) to offline calculate the optimal policy.

RL-Flooding: [87] is a constrained flooding routing technique, able to handle very dynamic WSN environments. Its basic idea consists of learning the costs to the single sink at each node in the network by using Q-Learning (very similar to the above proposed approaches). The costs can be defined as any metric based on hops, remaining energy on the nodes, energy required to reach the sink, etc. However, instead of using the single best route to route the packet, RL-Flooding allows all neighbors, whose cost to the sink is not greater than that of the last hop to forward the node. This makes the routing very flexible and the data delivery very fast and reliable. However, in more static and reliable environments it causes a lot of additional communication overhead.

Q-Fusion: A Q-Learning based routing protocol with somewhat different goal is presented in [89], where sensing nodes disseminate data tuples from military target detection applications to interested nodes. However, sinks or interested nodes are defined as forwarding nodes, able to fuse the data about some detected event e_i. The authors apply a Q-Learning technique, where the rewards (feedbacks) are defined for a data packet e_i as:

$$reward = \begin{cases} 1 & \text{if } b \text{ is a sink for } e_i \\ 0 & \text{if no reward arrived} \\ \max\limits_{n_b} Q^{t-1}(b) & \text{otherwise} \end{cases} \quad (29)$$

where b is the rewarding node with its Q-values at time $t-1$ with its neighbors n_b. Thus, the best fusion walks through the network are learnt, until the data packet reaches its TTL. There is no traditional sense of sinks or destinations in this application and it shows again the broad applicability of RL based techniques for learning any distributed goals in a network at nearly no communication overhead.

Q-PR: Q-Probabilistic Routing [47] is another variation of Q-Routing, where a geographic-based routing protocol is implemented for routing data from single source to a single sink. In each step of the algorithm to the sink a subset of neighboring nodes is selected to forward the packet. A complex coordination algorithm is implemented to make sure that even under highly lossy links the message is forwarded. The underlying approach is though based on Q-Learning, where the best neighbors are learnt through experience (in terms of geographic progress and residual energy). Compared to a non-RL based geographic routing protocol, Q-PR shows very good results, especially in terms of delivery rate.

RLGR: Another geographic-based routing protocol for WSNs based on reinforcement learning is RLGR [90]. The authors use RL to learn the best next hop to reach a single sink in a network, using a reward function which includes besides the traditional geographic progress to the sink also remaining energy on the nodes and automatic avoidance of void areas:

$$reward = \begin{cases} R & \text{if next hop is not sink} \\ R_C & \text{if next hop is sink} \\ -R_D & \text{if cannot find next hop (void area)} \\ -R_E & \text{if next hop has very low energy} \end{cases} \quad (30)$$

where $R = \beta \frac{pr}{pr_{avg}} + (1-\beta)\frac{E_r}{E_{initial}}$. Here the first term R combines normalized progress to the sink with normalized remaining energy; and the other three terms manage special cases. The protocol has been compared in simulation to GPSR, a well-known geographic based routing protocol for WSNs and has achieved very good results in terms of energy spreading and routing costs.

TPOT-RL-Routing: The authors of TPOT-RL (see Section III-F) have applied their algorithm to packet routing in a network in [50]. TPOT-RL should be usually used when many agents act as a team and have to learn to cooperate in some task, without really knowing what the others are actually doing. The goal of the paper is a proof-of-concept of the wide applicability of the algorithm rather than developing a high-performance routing protocol and thus does not provide any comparison to other approaches. Besides this, although application to network routing is possible, it is not the best representation of the problem for two reasons: first, it presents the network as *one* system with a large number of possible states, and second, it assumes that additionally for every packet sent to the sink, a backward packet is sent also back to the *sender* to compute how many hops it traveled and thus to deliver some reward to the learning agent. Both assumptions are valid, but do not optimally use the properties of a network (wireless or not).

SAMPLE: Collaborative RL (see Section III-F) is defined and applied to optimal routing in a mobile ad hoc network in [51]. The approach learns the optimal routing policy through feedback among agents, using Boltzmann exploration strategy. A routing protocol is implemented on top of collaborative RL,

called SAMPLE, and tested in different network topologies with various mobility. The approach is fully distributed and the implementation is feasible also for WSNs, since all routing information is sent together with the data packets.

FROMS: An energy-aware multicast routing protocol based on Q-Learning called FROMS is presented in [46]. Its goal is to minimize the energy spent in a network, while delivering packets to many sinks simultaneously. The idea is based on an optimal broadcast Steiner tree, where a minimum number of broadcasts are needed to deliver one packet from an independent source to all sinks.

Each sensor node is a learning agent, which learns the best hop costs to any combination of sinks in the network. Actions are possible next hop(s) for routing the packet towards the sinks. An important property of the learning mechanism is the separation of the possible actions into actions and sub-actions: a sub-action involves only one neighboring sensor, while a complete action involves the sub-actions to many neighboring sensors, so that the full desired set of sinks is reached. For more examples and details of this mechanism, please refer to the original paper.

The Q-values of the sub-actions are initialized based on the individual hop counts to each of the sinks, which are disseminated during an announcement message from each sink. This initial value is an upper bound estimate of the real costs:

$$Q(a_i) = \left(\sum_{d \in D_i} hops_d^{n_i} \right) - 2(\mid D_i \mid -1) \tag{31}$$

where D_i are the targeted destinations. The Q values of the complete actions are a sum of the Q values of the involved neighbors. At each step of the algorithm, a forwarding node selects an action to reach the desired set of destinations. This can be done greedily, randomly or uniformly and is referred to as the exploration strategy. Before broadcasting the packet to all the neighbors, the forwarding node also includes its best Q-value for this set of destinations, thus giving reward to its neighbors. When receiving a packet from a neighbor, a node has two procedures to follow: to compete its new Q-value given the included reward and to forward the packet, if applicable. Updating of the Q-values is done according to:

$$Q_{new}(a_i) = Q_{old}(a_i) + \alpha(R(a_i) - Q_{old}(a_i)) \tag{32}$$

where the reward is computed as

$$R(a_i) = c_a + MIN_a Q(a) \tag{33}$$

Simulation results show that FROMS performs well even in case of mobility or node failure and is able to achieve up to 25% energy savings, compared to a simple Directed Diffusion multicast approach. The exploration strategy, the initialization of the Q-values, as well as an eventual exploration stop greatly influence the properties of the protocol and its performance. Basically it can be tuned to perform best in different scenarios, but it requires deep knowledge of the protocol and some time for experimenting. Also, the mechanism shows its full power under long-lived data flows, so that the costly exploration phase stabilizes and near-optimal routes can be followed.

Clique: An attempt to efficiently solve the clustering problem in WSNs in presented in [86]. The most important disadvantage of traditional clustering algorithms is their high communication overhead for forming the clusters, and electing and announcing the cluster heads.

Clique [86] solves the problem by avoiding all-over the cluster head selection process. It assumes the nodes in the WSN have some a-priori clustering information, like a simple geographic grid or room or floor information in a building. It further assumes that the possibly multiple sinks in the network announce themselves through network-wide data requests. During the propagation of these requests all network nodes are able to gather 1-hop neighborhood information consisting of the remaining energy, hops to individual sinks and cluster membership. When data becomes available for sending, nodes start routing it directly to the sinks. At each intermediate node they take localized decisions whether to route it further to some neighbor or to act as a cluster head and aggregate data from several sources. Clique uses Q-Learning to select the best decision.

The learning agents are the nodes in the network. The available actions are $a_{n_i} = (n_i, D)$ with $n_i \in \{N, self\}$, or in other words either routing to some neighbor in the same cluster or serving as cluster head and aggregating data arriving from other nodes. After aggregation, Clique hands over the data packet control to the routing protocol, which sends it directly and without any further aggregation to the sinks. In contrast to the original Q-Learning, Clique initializes the Q-Values not randomly or with zeros, but with a initial estimation of the real costs of the corresponding routes, based on the hops to all sinks and the remaining batteries on the next hops. The update rules for the Q-Values are the original Q-Learning ones, where the learning constant is set to 1 to speed up the learning process. The reward is simply the best available Q-Value at the rewarding node plus the cost of sending to this node. The most important property of Clique is its role-free nature. In contrast to most cluster head selection algorithms, it does not try to find the optimal cluster head (in terms of cost), but incrementally *learns* the best without knowing either where or who the real cluster heads are. As a result, at the beginning of the protocol, multiple nodes in the cluster may act as cluster heads. While this temporarily increases the overhead, it is a short-term tradeoff in comparison to the overhead required to agree on a single cluster head. Later in the protocol operation, after the real costs have been learned, multiple cluster heads occur only in disconnected clusters, where a single cluster head cannot serve all cluster members.

VI. Scheduling, Data Aggregation and Sensor Fusion, and QoS Management

A. Scheduling

Several aspects of scheduling in WSNs are accomplished through CI techniques like fuzzy neural network, GA, AIS, RL and hybrids of PSO and GA. Table VI summarizes some CI based solutions to scheduling in WSN discussed below.

1) Fuzzy Neural Network: A fuzzy Hopfield neural network is proposed in [97] to solve the TDMA broadcast scheduling problem in WSNs. The problem is treated as discrete energy minimization problem that is mapped into a Hopfield neural network. The objective of the broadcast scheduling problem is to find a conflict-free transmission schedule for each node in TDMA slots. The optimization criterion is to find the TDMA schedule having minimal TDMA cycle length and maximum node transmissions. The results of fuzzy Hopfield neural network are compared against methods proposed earlier, namely, state vector coloring noisy chaotic neural network [105], Hopfield neural network-GA [106] and mean field annealing [107]. TDMA frame length, channel utilization factor, and average time delay are used as metrics for comparison. The paper portrays fuzzy Hopfield neural network's ability to minimize TDMA frame length and maximize the channel utilization in order to achieve the lower time broadcast delay. However, the paper compares the performance of fuzzy Hopfield neural network with those of other NN-based techniques reported earlier, but not the traditional scheduling methods. Moreover, the paper does take into account computational overheads.

2) Evolutionary Algorithms: GA is employed effectively in communication scheduling to minimize the effects of bio-heating from tissue implanted bio-sensor network in [98]. Tissues are sensitive to increase in temperature resulting from normal operation of sensor nodes. The cluster heads that transmit more frequently, or those at higher power level are likely to cause more damage to tissues. Therefore, it is necessary to rotate cluster head allocation in order to minimize thermal effects. The algorithm *GA Thermal Effect* reported in [98] uses GA for fast computation of optimal rotation sequence based on a parameter called temperature increase potential. It is assumed that all sensors are surgically implanted, and the base station uses the knowledge of precise locations of all sensors to compute cluster head rotation sequences. The study uses a rotation sequence as a chromosome. GA uses order crossover in which individuals are randomly chosen for mating, and offspring are produced by swapping randomly selected segments of parent's chromosomes. Mutation is performed by swapping two randomly selected positions in a chromosome sequence. Roulette wheel selection is used. The paper presents a comparison of the time taken to determine a minimum temperature leadership rotation sequence by GA and other methods that use finite-difference time-domain approach. It is shown that the GA is several orders faster than the other methods.

Dynamic Alliance GA: A dynamic alliance model based on GA is proposed for sleep scheduling of nodes in a randomly deployed large scale WSN in [99]. Such networks deploy a large number of redundant nodes for better coverage, and how to manage the combination of nodes for a prolonged network operation is a major problem. The scheme proposed in the article divides the network life into rounds. In each round, a set of nodes is kept active (this set is referred to as a dynamic alliance) and the rest of the nodes are put in sleep mode. It is ensured that the set of active nodes has adequate coverage and connectivity. When some of the active nodes die, blind spots appear. At this time, all nodes are woken up for a decision on the next set of nodes to remain active in the next round. This is clearly a multi-objective optimization problem. The first objective is to minimize the overall energy consumption of the dynamic alliance, and the second objective is to minimize the number of the nodes in the dynamic alliance.

For a random deployment of N nodes, chromosomes in the proposed GA are N-bit binary strings in which a binary 1 in position k represents that the node k is a part of the current dynamic alliance. Fitness function is obtained by goal programming [3]. GA uses roulette wheel selection, and a 2-point crossover with a cross over probability P_c such that $0.6 \leq P_c \leq 1$. Mutation is performed by flipping of a random bit with a probability of $P_m = 0.34$. The best 10 individuals of a generation and their 10 best offsprings go the next generation. When the change in fitness of consecutive generations converges within 0.01%, the GA is stopped.

The article presents simulations results on a random deployment of a 40-node network in an area of 100×100 m^2. However, although the algorithm works with random deployments, gathering the topology information on a single base station is critical and non-feasible in a realistic scenario.

Active Interval Scheduling: A scheduling problem called the active interval scheduling problem in hierarchical WSNs for long-term periodical monitoring is introduced in [100], and an approach based on compact GA is proposed as a solution thereto. A hierarchical sensor network has its nodes partitioned into several clusters, each controlled by a local control center. Each local control center collects detection results from nodes in its cluster, and forwards the same to a global control center.

[3]Goal programming is a technique used by decision-makers to solve multi-objectives decision-making problems to find a set of satisfying solutions [108].

TABLE VI
A SUMMARY OF CI APPLICATIONS IN WSN SCHEDULING SURVEYED IN SUBSECTION VI-A

CI Paradigm	Algorithm	Articles	Simulation/Real-deployment/Overview	Centralized/Distributed
Fuzzy neural network	FHNN-Scheduling*	[97]	Simulation	Centralized
EA	GA Thermal Effect*	[98]	Evaluation through simulation	Centralized
	Dynamic Alliance-GA*	[99]	Simulation	Centralized
	Active Interval Scheduling	[100]	Simulation	Centralized
AISs	Immune Based Scheduling*	[101]	Simulation	Centralized
RL	Actor-Critic	[102]	Simulation	Distributed
	RL-MAC	[103]	NS-2 simulation	Distributed
A Hybrid of PSO and GA	GA-PSO*	[104]	Simulation	Centralized

*This algorithm is renamed here for easy reference.

The sensing operation is divided into sensing slots. In each slot, all nodes in a cluster wake up at a preset time, record and broadcast sensor data, and go to sleep at a preset time decided by the local control center. Length of time for which the nodes remain active is called active interval. The active times of neighboring clusters have to be mutually exclusive, otherwise communication in one cluster interferes with that in another. The length of the active interval should be minimum in order to save power, but is should be long enough to give all sensor nodes a chance to transmit their data.

Article [100] uses compact GA to determine the active interval for each cluster in the network depending on the number of nodes in the cluster. Length of the active interval is taken as the fitness function that GA seeks to minimize. The GA chromosomes represent schedules. Appropriate competition operator and cross over operator are defined. Comparison between the results of compact GA and those of two greedy interval scheduling algorithms proposed by the same authors in [109] are presented. GA is shown to produce schedules having lesser time costs, which means lesser power consumed.

3) Artificial Immune Systems: An immune system based node scheduling scheme is presented for power minimization and collision preemption in [101]. The objective of the scheme are to optimize the route of each hop of N nodes to minimize power consumed, and to order the nodes in path ensuring collision free propagation. In the proposed method, antigen (problem domain) is represented as sets of $A(i)$, i=1,2,...,N, where i denotes serial number of a node, and $A(i)$ denotes its index. Antibody (solution space), is represented using array codes. The code length of antibody gene is same as the length of antigen code L. Each row denotes the set of $B_j(i)$, which is the code of the i^{th} node on the route from j^{th} source to the sink. Therefore, routes from N nodes to the sink consist of an array of length $N \times L$. Chromosomes are randomly initialized. Crossover and aberrance are the two immune operators defined. Crossover swaps two selected chromosomes, while Aberrance operator selects a node and makes a variation ensuring that all required conditions are met. This method is reported to outperform direct, multi-hop, static clustering and LEACH schemes.

4) Reinforcement Learning: Actor Critic Algorithm, a near optimal point-to-point communication framework based on RL is presented in [102]. The goal of the algorithm is to maximize throughput per total consumed energy in a sensor network, based on node-to-node communication. Given its current buffer size and last channel transmission gain, the node decides the best modulation level and transmit power to maximize the total throughput per consumed energy. For this, the authors use the standard RL algorithm and test their algorithm on a two-node and multi-node scenarios. Unfortunately no comparison to other state-of-the-art protocols is presented in order to evaluate the gain of the RL algorithm.

RL-MAC: [103] applies reinforcement learning to adjust the sleeping schedule of a MAC protocol in a WSN setting. The MAC protocol is very similar in its idea to the other WSN MAC protocols like S-MAC or T-MAC. It divides the time into frames and the frames into slots, where each node is allowed to transmit messages only during its own reserved slot. However, unlike other protocols, it changes the duration of the frames and slots according to the current traffic. At the beginning of its reserved slot, the node first transmits some control information, including also a reward for the other nodes. The reward function depends on the number of waiting messages on the nodes and on the number of successfully transmitted messages during the reserved slot. It reports higher data throughput and lower energy expenditure compared to S-MAC.

5) Hybrid Algorithms: Time division multiple access based medium access control is a popular technique used in sensor networks because it can reduce the delay, provide real-time guarantees and save power by eliminating collisions. It is customary in such a MAC to send the node in sleep mode when there is no need of transmission. But, unnecessary state transitions between the active and sleep modes lead to wastage of energy. A multi-objective TDMA scheduling method for many-to-one sensor networks is presented in [104]. The article proposes a hybrid of PSO and GA for optimization (therefore, the name *GA-PSO*). The performance of the hybrid algorithm is compared to that of PSO, max degree first coloring algorithm and node based scheduling algorithm. In TDMA scheduling problem, time is split into equal intervals called time slots. Each time slot accommodates a single packet between pairs of nodes in the network. The challenge here is to assign time slots to nodes so that collisions would not occur. The energy necessary in a N-node network is given by

$$EC = \sum_{i=1}^{N} [P_i^{tx} \times (t_i^{tx} - t_i^{s-tx}) + P_i^{rx} \times (t_i^{rx} - t_i^{s-rx})] \quad (34)$$

where P_i^{tx} and P_i^{rx} are power transmitted and received by the node i respectively. t_i^{tx} and t_i^{rx} are transmission and reception times at node i respectively; and t_i^{s-tx} and t_i^{s-rx} transition time between active and sleep modes respectively. The proposed hybrid optimization algorithm minimizes the objective function below.

$$F(s) = \alpha \times EC + (1 - \alpha) \times TS \quad (35)$$

where TS is the total number of slots under the schedule and α defines the trade off between the two objectives EC and TS.

A schedule is coded into a binary number using an encoding scheme. Given N tasks and each task i includes M_i hops, an individual is a sequence composed of all the task ID numbers from 0 to $N-1$, where each task number i appears for M_i times. Each individual has a length of $\sum_{i=0}^{N-1} M_i$ bits. GA uses tournament selection, one-point cross over and a two-position swap mutation. First, a random population is generated and the PSO is performed for a specific number of iterations. Then the GA is performed on the PSO optimized population until desired performance is obtained. GA is used to get closer to the optimal solution as is good at solving discrete problems. The results show that the schedule determined by hybrid algorithm in a 169 node, 489 link scenario 644 mJ of energy, which is

marginally better than the schedules determined by max degree first coloring algorithm and node based scheduling algorithm consume 740 mJ and 666 mJ respectively.

B. Data Aggregation and Sensor Fusion

Some CI based solutions to data aggregation and sensor fusion problem are discussed below and summarized in Table VII. The methods capable of automatic adjustment and self-adaptation are required for intelligent fusion of information from distributed nodes in a multi-sensor network. Data aggregation and sensor fusion are addressed through GA, fuzzy logic, RL and NNs. Some of the approaches focus on determining efficient aggregation paths and others deal with fusion rules. GAs have resulted in efficient data aggregation routes in a mobile agent based WSN because of their inherent parallel nature and ability to deal with difficult real world problems like non-homogeneous, noisy, incomplete and/or obscured information. However, these parallel search algorithms require the costs of collecting required information at a base station before determining the aggregation path for the agent. Due to its ability to converge quickly to the optimal solution, PSO proves to be a good substitute for GA. Similarly, ACO can be a good substitute for GA because of its distributed nature, which obviates collection of prior information at the base station before determining an optimal route. On the other hand, a neural network's ability to learn and dynamically adapt to the changing scenarios makes it a natural choice for information fusion applications. Reinforcement learning has also been applied successfully for solving the optimal aggregation path problem - it is fully distributed and able to adapt quickly to topology changes or failing nodes. Many applications, especially for monitoring or predicting events, require centralized gathering of data. In such applications, a centralized approach like neural networks, GA or PSO can be successfully applied to learn the properties of the data.

1) Fuzzy Logic: A novel distributed approach based on fuzzy numbers and weighted average operators to perform an energy efficient flooding-based aggregation is proposed in [110]. In this study, each sensor node maintains an estimate of the aggregation value represented as a symmetric triangular fuzzy number. Aggregation is done at each node if either a new measurement value is locally available to the node, or if a new value is received from a neighboring node. Each node maintains a table of sensor values received from neighboring nodes. Based on the estimate, a node decides if a newly measured sensor data has to be propagated in the network or not. This reduces the number of messages transmitted, and thus reduces the energy spent. The article presents the results of experiments on a network of twelve motes that use flooding-based aggregation, deployed in an apartment to monitor temperature profile over 24 hours. The article reports a reduced number of received and transmitted messages leading to network life-time of an impressive 418 days. However, no comparison to "normal" network lifetime is provided, which complicates the interpretation of their results.

2) Evolutionary Algorithms: Article [111] gives an overview of basic and advanced concepts, models, and variants of GA in various applications in information fusion.

GAgent: The issue of data aggregation for a target detection application is addressed in [112] through mobile agent-based distributed sensor networks wherein a mobile agent selectively visits the sensors and incrementally fuses the appropriate measurement data. GA is used to determine the optimal route for the agent to traverse (thus, the name *GAgent*). Mobile agents are special programs that are dispatched from a source node to be executed at remote nodes. A mobile agent arrives at a remote node, gains access to services and data, collects needed information or performs certain actions, and departs with the results. The number of nodes on the route and the sequence in which they are visited by a mobile agent have an impact on energy consumed, path loss, and detection accuracy. The energy consumed and path loss need to be minimized, while the signal energy has to be maximized.

The study uses a two-level chromosome representation, where the first level stands for the sequence of nodes in the route and the second level represents presence and absence of nodes in the route, as shown in Figure 8. Reproduction operator ensures that an offspring that has better fitness than the minimum in the current population remains in the population, and the parent having minimum fitness is eliminated. Two point crossover with big crossover probability (> 0.9) is used. Two mutation points are selected randomly and the values of these two points are exchanged. In addition, two mutation points are selected randomly and the order of nodes between these two points is reversed. Several case-studies with node sizes from 200 to 1500 are made. Results are compared with those of popular heuristic algorithms local-closest-first and global-closest-first. The results show that the GA comes up routes superior to the ones determined by both the aforementioned heuristic algorithms in all case studies.

In [112], the cost of gathering the information on a central unit to compute the optimal path is not considered. This cost does not apply to global-closest-first and local-closest-first distributed heuristic algorithms. The protocol still needs to be evaluated in a sophisticated network scenario, where all necessary costs are taken into account and compared.

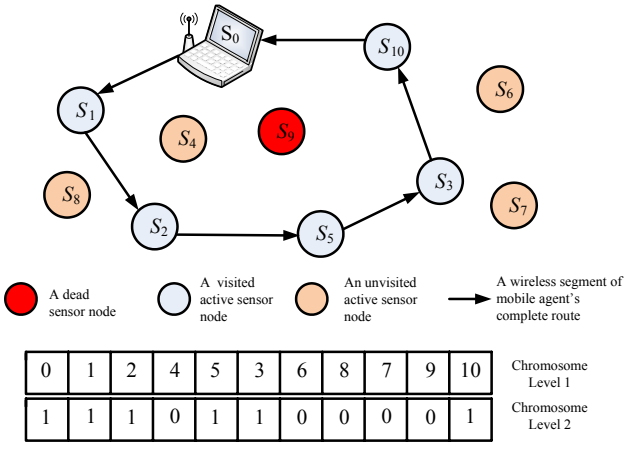

Fig. 8. A mobile agent's route in the distributed WSN scenario and the two-level chromosome code for the route

TABLE VII
A SUMMARY OF CI APPLICATIONS FOR DATA AGGREGATION AND SENSOR FUSION IN WSNS SURVEYED IN SUBSECTION VI-B

CI Paradigm	Algorithm	Articles	Simulation/Real-deployment/Overview	Centralized/Distributed
Fuzzy Logic	Flooding-Based Aggregation	[110]	Real deployment	Distributed
EA		[111]	Overview	
	GAgent*	[112]	Simulation	Centralized
	Stochastic Gradient GA	[113]	Simulation	A combination of centralized and distributed
	Parallel GA	[114]	Simulation	A combination of centralized and distributed
RL	Q-RC	[44]	Simulation	Distributed

*This algorithm is renamed here for easy reference.

GA-Stochastic Gradient and Parallel GA: GA is used in [113] and [114] to address the problem of optimal detection performance in sensor networks that have communication constraints. Both the studies consider the parallel fusion architecture in which local sensors feed their quantized decisions to a single fusion center through the medium that has a bit rate constraint. N sensors gather T measurements $y_{n,t}$ of a state of nature H per sensor, make a local decision $u_{n,t}$ per measurement using a mapping function γ^n and send local decisions to a single fusion center. The fusion center makes a binary global decision $\tilde{H} \in \{H_0, H_1\}$ based on the collection of local decisions from all sensors. Each sensor classifies each measurement into $L = 2^b$ classes, where b is the number of bits transmitted per measurement. The fusion center maps NT local decisions into one of the two classes H_0 and H_1 using a mapping function γ^0. It is assumed that each sensor has a quantizer characterized by $L-1$ thresholds defined in (36).

$$u_n = \begin{cases} 0 & \text{if } y_n \leq \lambda_{n,1} \\ 1 & \text{if } \lambda_{n,1} \leq y_n \leq \lambda_{n,2} \\ \vdots & \vdots \\ L-1 & \text{if } y_n > \lambda_{n,L-1} \end{cases} \tag{36}$$

Both articles [113] and [114] address the following issues:

1) Structure of the fusion rule that optimizes the detection probability error $P_e = \Pr(\tilde{H} \neq H)$.
2) Tradeoff between number of sensors N, number of bits per measurement b and SNR.
3) Convergence of the error decay rate as $N \to \infty$

In the stochastic gradient GA approach proposed in [113], fusion rules are represented in binary chromosome strings as follows: There are N sensors and each sensor classifies its measurement into L classes, a fusion rule is represented as a string of L^N bits. Each combination of local decisions is represented by a vector of N integers. Because $L = 2^b$, a combination of local decisions is represented as a string of bN bits. The approach proposed here uses GA to search for the optimal fusion rule and a gradient-based algorithm optimizing the local thresholds. The article reports good convergence of GA to the optimal solution. However, for each solution candidate of fusion rule in GA, $N(L-1)$ variables need to be evaluated for gradient.

In the parallel GA approach proposed in [114], GA is used to optimize both the fusion rule and the local thresholds simultaneously. Each chromosome here has a fusion rule and a set of local thresholds. Because the complex gradient calculations are not necessary, this approach has an advantage over stochastic gradient GA in terms of computational expenses. Results indicate that the parallel GA approach achieves the same detection error probability as the stochastic gradient GA does in homogeneous and heterogeneous sets of nodes, but its computational complexity is one order lesser.

3) Reinforcement Learning: As mentioned above, RL has been successfully applied to learning the best aggregation and routing path to the sink in a distributed manner, like the Q-RC protocol [44], which has been presented in Section V.

C. QoS Management

Table VIII lists the CI applications in QoS management discussed here. A holistic QoS approach has not received significant research attention, and energy efficiency has been the main QoS metric in research efforts. The existing QoS approaches are classified into stateful and stateless approaches depending on the mechanisms used to support QoS. The stateful approaches need flow state information to be maintained in the network nodes. This is difficult to manage in dynamic networks and it restricts the scalability. Stateless approaches typically use feedback-based mechanisms and local control to support service differentiation and real-time services. The stateless source-based approaches may get an old view of the real status of resources which can lead to the admission of more traffic than what an intermediate node can really support.

1) Fuzzy logic: A fuzzy logic based congestion estimation method within a proposed QoS architecture Fuzzy-QoS is presented in [115]. The architecture comprises of a QoS Management and Control module which is implemented both at node level and at the sink for a system level QoS administration. The Fuzzy-QoS approach uses packet arrival rate, buffer size in a sensor node to determine and maintain a fuzzy table for the conditions of buffer congestion. The fuzzy set consists of two fuzzy variables: packet arrival rate (p) and buffer size (s), $\vec{A} = \{p, s\}$. Each of these fuzzy variables takes values of $0, 0.5, 1$ depending on low, medium or high congestion conditions. Then, the fuzzy table is computed using max-min composition. This table denotes the estimation of the current congestion level. This helps in making decision regarding if a packet needs to be dropped. The article reports that the approach reduces the number of dropped important event-driven packets in comparison to that in continuous traffic and query-driven traffic models.

The approach named *Fuzzy-Control of Congestion* proposed in [116] is a fuzzy logic technique for improving the control of congestion through buffer threshold management in wireless ad hoc networks. The threshold function has a significant

TABLE VIII
A SUMMARY OF CI APPLICATIONS IN WSN QOS MANAGEMENT SURVEYED IN SUBSECTION VI-C

CI Paradigm	Algorithm	Articles	Simulation/Real-deployment/Overview	Centralized/Distributed
Fuzzy logic	Fuzzy-QoS*	[115]	Simulation	Combination of centralized and distributed
	Fuzzy-Control of Congestion	[116]	Simulation	Distributed
	FuzzyMARS	[117]	Simulation	Distributed
*This algorithm is renamed here for easy reference.				

influence on the performance of a network in terms of a average packet delay and throughput. Therefore, the selection of a particular threshold is critical for an adequate congestion control.

The fuzzy logic develops a more realistic representation of buffer occupancy that helps to offer an efficient decision making regarding if an in-coming packet should be accepted or rejected by a node. the definition of buffer occupancy will consider the two fuzzy cases of getting full and not getting full, rather than admit and no-admit in the previously reported approaches. This fuzzy representation replaces the two discrete sets by a continuous set membership, and performs small gradual transitions between different states of buffer occupancy. The sigmoidal membership function proposed here aims to determine the fuzzy threshold depending on the fullness of the buffer. The admit membership function is inversely proportional to the occupancy fullness level of buffer. Thus, when the occupancy fullness is small, the value of the admit membership function is big, and vice versa. Fuzzy rules are framed in such a way that when the value of the admit membership function is big, then the accepted incoming packets into buffer is increased; and when the value of the admit membership function is small, then the accepted incoming packets into buffer is reduced.

The article presents results of NS-2 simulation under diverse mobility and traffic conditions. In terms of traffic scalability, the Fuzzy-Control of Congestion achieves a reduction in terms of average end-to-end delay by about 74-92% in comparison to the IEEE 802.11 wireless networks, with almost the same throughput.

FuzzyMARS: A fuzzy logic QoS approach for wireless ad hoc network, named FuzzyMARS is proposed in [117]. The approach uses fuzzy logic for best-effort traffic regulation, real-time traffic regulation and admission control. The feedback delay from MAC layer is the input to the fuzzy logic, and traffic regulation rate is the output. The input and output variables are fuzzified into in three classes *low*, *medium*, and *high*, and fuzzy rules are defined. The mean-of-maxima method [34] for defuzzification. The performance evaluation results show that FuzzyMARS experiences low and stable delays under different channel conditions, traffic scalability, and network mobility while preserving the throughput.

VII. GUIDE TO CI METHODS FOR WSNS

Many CI methods have outperformed or complimented conventional methods under uncertain environments and severe limitations in power supply, communication bandwidth, and computational capabilities. However, all works presented here are not the best possible solutions and many have not been compared to traditional or to other CI approaches. Additionally, only a few researchers have evaluated their algorithms under real WSN environments like test-bed or deployments.

Authors' findings have been summarized in Figure 9. The columns of the table represent the application areas in WSNs considered in this survey, while the rows represent the main CI techniques. The size of the black circles represent the number of articles surveyed in this paper for the particular combination of WSN problem and CI approach. In contrast, the shadowing of the box represents an evaluation of the applicability and suitability of the CI method for the particular problem. Of course, this evaluation is not always true: It depends highly on the exact CI algorithm, its parameters and the exact formulation of the problem. However, this overview gives a good insight about which CI method to explore first, when trying to solve a WSN problem.

Design and deployment is usually a centralized problem, where an optimal architecture for the WSN to be deployed is determined. Here, centralized models like NNs, GAs and PSO are very well suited. They can produce optimal results from large data-sets and memory and processing restrictions do not apply. RL can be used for online optimal deployment, like optimizing current deployments or even self-deployment with mobile robots.

For *localization*, it looks like NNs and GAs are the best suited techniques, although they need to be used in a centralized manner. The problem is the high variance of the localization data, like RSSI values to compute distances between nodes. Here, large data-sets are needed to produce good localization results, which can be handled only centrally anyway.

Fuzzy logic is well suited for *security* and *QoS* problems. It is able to compute general non-optimal rules that can accommodate larger variance of the data, as in case of security applications.

Routing and clustering seems to be the most popular WSN problem for applying CI methods (in fact, it is also a very active research area in general). However, not all CI methods are equally suited. NNs and EAs have very high processing demands and are usually centralized solutions. In the case of NNs, learning can be also be conducted online at each of the nodes, but is slow and has high memory requirements. These two CI approaches are slightly better suited for clustering when the clustering schemes can be pre-deployed (see also the following discussion of design and deployment issues). Fuzzy logic is very well suited for implementing routing and clustering heuristics and optimizations, like link or cluster head quality classification. However, it generates non-optimal solutions and fuzzy rules need to be re-learnt upon topology changes.

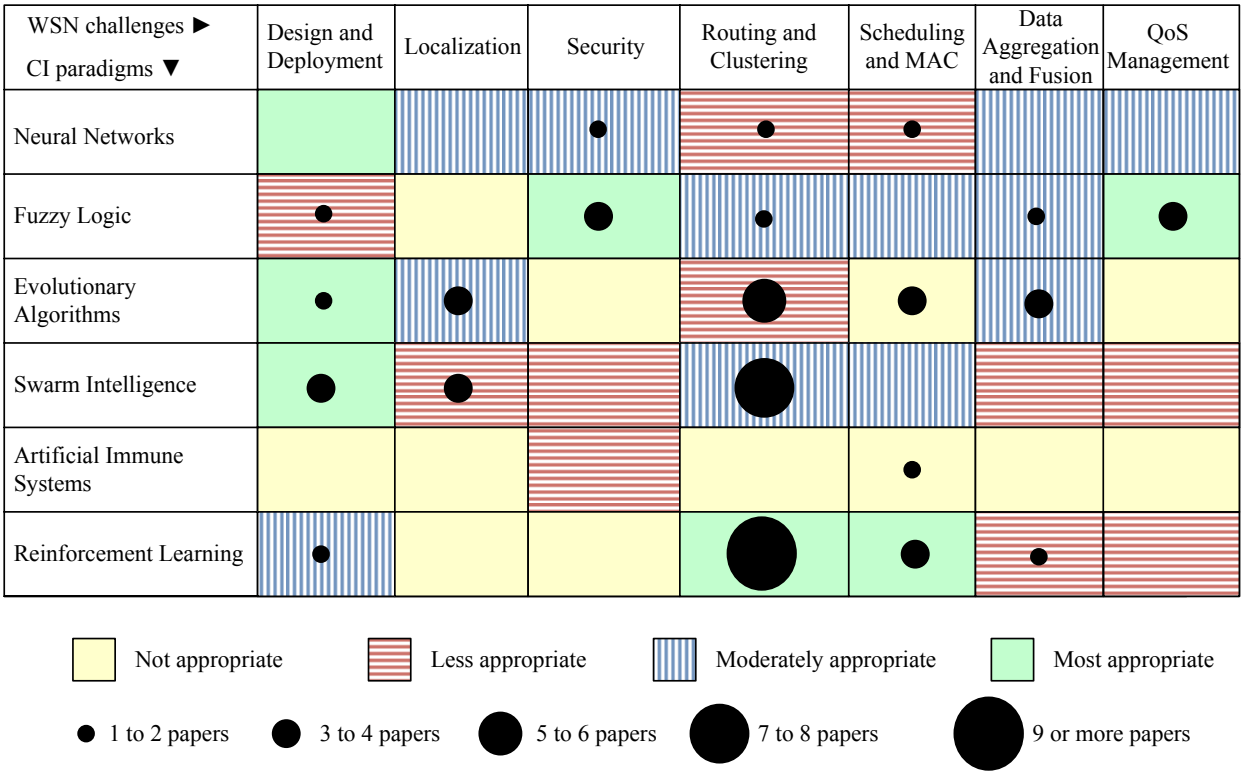

Fig. 9. An overview of WSN challenges and the CI paradigms applied to address them

SI is a very popular paradigm for computing routing schemes for MANETs. However, in WSNs it requires high communication overhead for sending ants separately for managing the routes. Besides, SI usually requires the ants to be sent back to the source of the data, which further increases the energy expenditure. The SI model needs to be changed to accommodate the WSNs requirements and properties, but this has not been done so far.

RL is the best option when dealing with distributed and dynamic problems like routing and clustering for WSNs. It has exactly the needed properties, and it has been applied very successfully to both problems, even on real test-bed. RL produces optimal routing decisions, it is flexible and robust against node and link failures and it is fully distributed. Its communication requirements are nearly zero and it maintains data delivery even in case of topology changes. Q-Learning is the most popular RL method to apply to WSNs because it has the lowest communication and processing demands and its model is very flexible.

Nearly the similar properties and discussion as for routing and clustering hold good for *scheduling and MAC* problems. Scheduling is also a highly distributed and dynamic problem, and again RL is the best suited technique. Fuzzy logic can be used for implementing heuristics and optimizations. SI can be applied too, but with higher communication requirements.

When dealing with *data aggregation and fusion*, the best suited CI methods are fuzzy logic, evolutionary algorithms and neural networks. However, are centralized approaches, which increases the communication overhead in the network while gathering the data on a central node for processing and disseminating the results back to the nodes. On the other hand, data fusion is very often accomplished only after gathering the data on the base station or on cluster heads, which makes the approaches well suited. All other CI methods are rather secondary choices, since their models do not correspond well to the problem.

It is interesting to note that two CI techniques, NNs and AIS, have been rarely applied to WSNs. This is particularly awkward in the case of NNs, because this paradigm is very well studied and there exist many different NN models with different properties. AISs are rather a new research area and are not so well studied. They could be good solutions to security problems, but a general evaluation remains open to future research.

VIII. Conclusions and Future Applications of CI in WSNs

Recent literature shows that researchers have focused their attention on innovative use of CI techniques to address WSN issues such as design and deployment, localization, security, optimal routing and clustering, scheduling, data aggregation and fusion, and QoS management. Recent implementations of CI methods in various dynamical and heterogeneous networks are presented in this survey paper. CI paradigms and numerous challenges in sensor networks are briefly introduced, and the CI approaches used by researchers to address the challenges are briefly explained. In addition, a general evaluation of CI algorithms is presented, which will serve as a guide for using CI algorithms for WSNs.

An advanced CI approach called adaptive critic design holds promise to generate practical optimal/sub-optimal approaches to the distributed sensor scheduling problem. There are successful applications of adaptive critic designs in power systems, which show that the technique provides guaranteed stable optimal solutions under uncertainties and noise [118]. The potential of adaptive critic designs remains to be exploited in the field of WSN scheduling.

In addition, there is a vast scope for research in methods for efficient data representation and advanced algorithms for data reduction. Currently there are no techniques for distributed estimation under dynamic communication constraints. Existing centralized multi-sensor estimation techniques assume that the choice of which data to send from any particular sensor to the centralized node is fixed. This approach suffers serious disadvantages and is typically feasible for certain non-critical static sensor situations.

Future research is likely to focus on developing a well founded analytical approach to distributed multi-sensor estimation problem where there are time varying communication bandwidth constraints.

Cross-layer design and parameter learning is envisioned to be an interesting new research area for CI in WSNs. Right now, the majority of the solutions presented here apply CI to one limited problem in areas like multicast routing, link quality, optimal clustering or placement. However, most issues arise from cross-layer incompatibility and the high human intervention needed for parameter setting and adjustment. Learning platforms and paradigms are needed rather than specialized solutions.

In spite of a multitude of successful CI applications in WSNs, the main concern is that the most of these algorithms or protocols are still in development stage, and they may remain forever in non-finalized state. Very few protocols have grown out of the simulation environment. Most of them do not even consider unreliable or asymmetric links, node failure and mobility. Besides, a common problem is the lack of comparison to conventional state-of-the-art protocols to clearly identify the advantages of introducing CI. Thus, the goal of CI research community for the future of CI in WSNs is to improve already existing solutions, refine them and define well-performing real-world protocols. There are only a few already published initiatives in this direction.

REFERENCES

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

[2] C. Y. Chong and S. Kumar, "Sensor networks: Evolution, opportunities, and challenges," *Proc. IEEE*, vol. 91, no. 8, pp. 1247–1256, Aug. 2003.

[3] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh, "Deploying a wireless sensor network on an active volcano," *IEEE Internet Computing*, vol. 10, no. 2, pp. 18–25, 2006.

[4] K. Martinez, P. Padhy, A. Riddoch, R. Ong, and J. Hart, "Glacial Environment Monitoring using Sensor Networks," in *Proc. 1st Workshop on Real-World Wireless Sensor Networks (REALWSN)*, Stockholm, Sweden, 2005, p. 5pp.

[5] I. Talzi, A. Hasler, S. Gruber, and C. Tschudin, "Permasense: investigating permafrost with a WSN in the swiss alps," in *Proc. 4th Workshop on Embedded Networked Sensors (EmNets)*, Cork, Ireland, 2007, pp. 8–12.

[6] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, "A taxonomy of wireless micro-sensor network models," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 6, no. 2, pp. 28–36, 2002.

[7] K. Langendoen, A. Baggio, and O. Visser, "Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture," in *Proc. 20th Int. Symposium on Parallel and Distributed Processing Symposium (IPDPS)*, Rhodes Island, Greece, 2006.

[8] J. McCulloch, P. McCarthy, S. M. Guru, W. Peng, D. Hugo, and A. Terhorst, "Wireless sensor network deployment for water use efficiency in irrigation," in *Proc. conf. Workshop on Real-world Wireless Sensor Networks (REALWSN)*, Glasgow, Scotland, 2008, pp. 46–50.

[9] E. A. Basha, S. Ravela, and D. Rus, "Model-based monitoring for early warning flood detection," in *Proc. conf. 6th ACM conf. on Embedded network sensor systems (SenSys)*, New York, NY, USA, 2008, pp. 295–308.

[10] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli, "The hitchhiker's guide to successful wireless sensor network deployments," in *Proc. 6th ACM conf. on Embedded network sensor systems (SenSys)*, New York, NY, USA, 2008, pp. 43–56.

[11] T. Naumowicz, R. Freeman, A. Heil, M. Calsyn, E. Hellmich, A. Braendle, T. Guilford, and J. Schiller, "Autonomous monitoring of vulnerable habitats using a wireless sensor network," in *Proc. 3rd Workshop on Real-World Wireless Sensor Networks (REALWSN)*, Glasgow, Scottland, 2008, pp. 51–55.

[12] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler, "Lessons From a Sensor Network Expedition," in *Proc. 1st European Workshop on Sensor Networks (EWSN)*, Berlin, Germany, 2004, pp. 307–322. [Online]. Available: http://www.cs.berkeley.edu/~polastre/papers/ewsn04.pdf

[13] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[14] E. Cayirci and T. Coplu, "SENDROM: sensor networks for disaster relief operations management," *Wireless Networks*, vol. 13, no. 3, pp. 409–423, 2007.

[15] G. Wittenburg, K. Terfloth, F. López Villafuerte, T. Naumowicz, H. Ritter, and J. Schiller, "Fence monitoring – experimental evaluation of a use case for wireless sensor networks," in *Proc. 4th European Conf. on Wireless Sensor Networks (EWSN)*, Delft, The Netherlands, 2007, pp. 163–178.

[16] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: an acquisitional query processing system for sensor networks," *ACM Trans. on Database Systems*, vol. 30, no. 1, pp. 122–173, 2005.

[17] N. Patwari, J. Ash, S. Kyperountas, A. Hero, R. Moses, and N. Correal, "Locating the nodes: Cooperative localization in wireless sensor networks," *IEEE Signal Processing Mag.*, vol. 22, no. 4, pp. 54–69, July 2005.

[18] J. Aspnes, T. Eren, D. Goldenberg, A. Morse, W. Whiteley, Y. Yang, B. Anderson, and P. Belhumeur, "A theory of network localization," *IEEE Transactions on Mobile Computing*, vol. 5, no. 12, pp. 1663–1678, Dec. 2006.

[19] A. Boukerche, H. Oliveira, E. Nakamura, and A. Loureiro, "Localization systems for wireless sensor networks," *IEEE Wireless Commun. Mag.*, vol. 14, no. 6, pp. 6–12, December 2007.

[20] I. Guvenc and C.-C. Chong, "A survey on TOA based wireless localization and NLOS mitigation techniques," *IEEE Commun. Surveys Tuts.*, vol. 11, no. 3, pp. 107–124, 2009.

[21] R. R. Brooks and S. S. Iyengar, *Multi-sensor fusion: Fundamentals and applications with software*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1998.

[22] R. Rajagopalan and P. Varshney, "Data-aggregation techniques in sensor networks: a survey," *IEEE Commun. Surveys Tuts.*, vol. 8, no. 4, pp. 48–63, Fourth Quarter 2006.

[23] P. Jiang, Y. Wen, J. Wang, X. Shen, and A. Xue, "A study of routing protocols in wireless sensor networks," in *Proc. 6th World Congress on Intelligent Control and Automation WCICA 2006*, Y. Wen, Ed., vol. 1, 2006, pp. 266–270.

[24] F. Hu and N. K. Sharma, "Security considerations in ad hoc sensor networks," *Ad Hoc Networks*, vol. 3, no. 1, pp. 69–89, 2005.

[25] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, vol. 1, no. 2–3, pp. 293–315, Sep 2003.

[26] X. Chen, K. Makki, K. Yen, and N. Pissinou, "Sensor network security: a survey," *IEEE Commun. Surveys Tuts.*, vol. 11, no. 2, pp. 52–73, 2009.

[27] D. Chen and P. Varshney, "QoS support in wireless sensor networks: A survey," June 2004.

[28] G. K. Venayagamoorthy, "A successful interdisciplinary course on computational intelligence," *IEEE Computational Intelligence Magazine*, vol. 4, no. 1, pp. 14–23, 2009.

[29] A. Engelbrecht, *Computational Intelligence: An Introduction*, 2nd ed. New York, USA: John Wiley & Sons, 2007.

[30] A. Konar, *Computational Intelligence: Principles, Techniques and applications*. Springer, 2005.

[31] S. Haykin, *Neural networks: A comprehensive foundation*. Prentice Hall, 1994.

[32] P. Baldi and K. Hornik, "Learning in linear neural networks: A survey," *IEEE Trans. Neural Networks*, vol. 6, no. 4, pp. 837–858, 1995.

[33] L. A. Zadeh, "Soft computing and fuzzy logic," *IEEE Trans. Software Eng.*, vol. 11, no. 6, pp. 48–56, 1994.

[34] L. A. Zadeh, "Fuzzy logic = computing with words," *IEEE Trans. Fuzzy Syst.*, vol. 4, no. 2, pp. 103–111, May 1996.

[35] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. on Neural Networks*, vol. 4, 27 Nov.–1 Dec. 1995, pp. 1942–1948.

[36] M. Dorigo, "Optimization, learning and natural algorithms," Ph.D. dissertation, Politecnico di Milano, Italy, 1992.

[37] C. Tovey, "The honey bee algorithm: A biological inspired approach to internet server optimization," in *the Alumni Magazine for ISyE at Georgia Institute of Technology*, Spring 2004, pp. 13–15. [Online]. Available: http://ormstomorrow.informs.org/archive/fall04/Tovey%20article.pdf

[38] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *EP '98: Proc. 7th Int. Conf. on Evolutionary Programming VII*. London, UK: Springer-Verlag, 1998, pp. 591–600.

[39] Y. del Valle, G. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, and R. Harley, "Particle swarm optimization: Basic concepts, variants and applications in power systems," *IEEE Trans. Evol. Comput.*, vol. 12, no. 2, pp. 171–195, April 2008.

[40] D. Dasgupta, "Advances in artificial immune systems," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 40–49, Nov. 2006.

[41] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, March 1998.

[42] L. P. Kaelbling, M. L. Littman, and A. P. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[43] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, Cambridge University, Cambridge, England, 1989.

[44] P. Beyens, M. Peeters, K. Steenhaut, and A. Nowe, "Routing with compression in wireless sensor networks: A Q-learning approach," in *Proc. 5th European Workshop on Adaptive Agents and Multi-Agent Systems (AAMAS)*, 2005.

[45] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," *Advances in Neural Information Processing Systems*, vol. 6, 1994.

[46] A. Förster and A. L. Murphy, "FROMS: Feedback routing for optimizing multiple sinks in WSN with reinforcement learning," in *Proc. 3rd Int. Conf. on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2007.

[47] R. Arroyo-Valles, R. Alaiz-Rodrigues, A. Guerrero-Curieses, and J. Cid-Suiero, "Q-probabilistic routing in wireless sensor networks," in *Proc 3rd Int. Conf. on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2007.

[48] S. Kumar and R. Miikkulainen, "Dual reinforcement Q-routing: An on-line adaptive routing algorithm," in *Proc. Artificial Neural Networks in Engineering Conf.*, 1997.

[49] P. Stone and M. Veloso, "Team-partitioned, opaque-transition reinforcement learning," in *Proc. 3rd annual conf. on Autonomous Agents (AGENTS)*. New York, NY, USA: ACM Press, 1999, pp. 206–212.

[50] P. Stone, "TPOT- RL applied to network routing," in *Proc. 17th Int. Conf. on Machine Learning*. San Francisco, CA: Morgan Kaufmann, 2000.

[51] J. Dowling, E. Curran, R. Cunningham, and V. Cahill, "Using feedback in collaborative reinforcement learning to adaptively optimize MANET routing," *IEEE Trans. Syst., Man, Cybern. B*, vol. 35, no. 3, pp. 360–372, 2005.

[52] L. Zhao and Q. Liang, "Fuzzy deployment for wireless sensor networks," in *Proc. IEEE Int. Conf. on Computational Intelligence for Homeland Security and Personal Safety CIHSPS*, Q. Liang, Ed., 2005, pp. 79–83.

[53] N. B. B. Jessica A. Carballido, Ignacio Ponzoni, "CGD-GA: A graph-based genetic algorithm for sensor network design." *Inf. Sci.*, vol. 177, no. 22, pp. 5091–5102, 2007.

[54] P. Ngatchou, W. Fox, and M. El-Sharkawi, "Distributed sensor placement with sequential particle swarm optimization," in *Proc. IEEE Swarm Intelligence Symposium SIS*, 8–10 June 2005, pp. 385–388.

[55] C. Mendis, S. Guru, S. Halgamuge, and S. Fernando, "Optimized sink node path using particle swarm optimization," in *Proc. 20th Int. Conf. on Advanced Information Networking and Applications AINA*, S. Guru, Ed., vol. 2, 2006.

[56] J. Hu, J. Song, X. Kang, and M. Zhang, "A study of particle swarm optimization in urban traffic surveillance system," in *Proc. IMACS Multiconference on Computational Engineering in Systems Applications*, J. Song, Ed., vol. 2, 2006, pp. 2056–2061.

[57] M. Seah, C. Tham, K. Srinivasan, and A. Xin, "Achieving coverage through distributed reinforcement learning in wireless sensor networks," in *Proc. 3rd Int. Conf. on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2007.

[58] S. Gonzalez-Valenzuela, S. T. Vuong, and V. C. M. Leung, "A reinforcement-learning approach to service directory placement in wireless ad hoc networks," in *Proc. IEEE 5th Int. Workshop on Aplications and Services on Wireless Networks*. IEEE Press, 2005.

[59] J. Travers and S. Milgram, "An experimental study of the small world problem," *Sociometry*, vol. 32, no. 4, pp. 425–443, 1969.

[60] G.-F. Nan, M.-Q. Li, and J. Li, "Estimation of node localization with a real-coded genetic algorithm in WSNs," in *Proc. Int. Conf. on Machine Learning and Cybernetics*, vol. 2, 2007, pp. 873–878.

[61] M. Marks and E. Niewiadomska-Szynkiewicz, "Two-phase stochastic optimization to sensor network localization," in *Proc. Int. Conf. on Sensor Technologies and Applications SensorComm*, 2007, pp. 134–139.

[62] Q. Zhang, J. Huang, J. Wang, C. Jin, J. Ye, W. Zhang, and J. Hu, "A two-phase localization algorithm for wireless sensor network," in *Proc. Int. Conf. on Information and Automation ICIA*, 2008, pp. 59–64.

[63] A. Gopakumar and L. Jacob, "Localization in wireless sensor networks using particle swarm optimization," in *Proc. IET Int. Conf. on Wireless, Mobile and Multimedia Networks*, 2008, pp. 227–230.

[64] R. V. Kulkarni, G. Venayagamoorthy, and M. X. Cheng, "Bio-inspired node localization in wireless sensor networks," in *Proc. IEEE Int. conf. on Systems, Man and Cybernetics*, Oct. 2009.

[65] H. Wolkowicz, *Handbook of Semidefinite Programming*. Kluwer Academic Publishers, 2000.

[66] P. J. M. Laarhoven and E. H. L. Aarts, Eds., *Simulated annealing: Theory and applications*. Norwell, MA, USA: Kluwer Academic Publishers, 1987.

[67] K. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Syst. Mag.*, vol. 22, no. 3, pp. 52–67, 2002.

[68] R. V. Kulkarni and G. Venayagamoorthy, "Neural network based secure media access control protocol for wireless sensor networks," in *Proc. Int. Joint Conf. on Neural Networks*, June 2009.

[69] R. V. Kulkarni, G. K. Venayagamoorthy, A. V. Thakur, and S. K. Madria, "Generalized neuron based secure media access control protocol for wireless sensor networks," in *Proc. IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making MCDM*, Mar 30-April 2 2009, pp. 16–22.

[70] Q. Ren and Q. Liang, "Secure media access control (MAC) in wireless sensor networks: Intrusion detections and countermeasures," in *Proc. 15th IEEE Int. Symposium on Personal, Indoor and Mobile Radio Communications PIMRC*, vol. 4, 5–8 Sept. 2004, pp. 3025–3029.

[71] Q. Ren and Q. Liang, "Fuzzy logic-optimized secure media access control (FSMAC) protocol wireless sensor networks," in *Proc. IEEE Int. Conf. on Computational Intelligence for Homeland Security and Personal Safety CIHSPS*, March 31 –April 1 2005, pp. 37–43.

[72] A. Förster, "Machine learning techniques applied to wireless ad hoc networks: Guide and survey," in *Proc. 3rd Int. Conf. on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2007.

[73] M. Di and E. Joo, "A survey of machine learning in wireless sensor networks," *Proc. 6th Int. Conf. on Information, Communications and Signal Processing*, 2007.

[74] J. Barbancho, C. León, J. Molina, and A. Barbancho, "Giving neurons to sensors: QoS management in wireless sensors networks." in *Proc. IEEE Conf. on Emerging Technologies and Factory Automation ETFA*, C. Leon, Ed., 2006, pp. 594–597.

[75] I. Gupta, D. Riordan, and S. Sampalli, "Cluster-head election using fuzzy logic for wireless sensor networks," in *Proc. 3rd Annual Communication Networks and Services Research Conf.*, D. Riordan, Ed., 2005, pp. 255–260.

[76] M. Islam, P. Thulasiraman, and R. Thulasiram, "A parallel ant colony optimization algorithm for all-pair routing in MANETs," in *Proc. Int. Parallel and Distributed Processing Symposium*, P. Thulasiraman, Ed., 2003.

[77] S. Wazed, A. Bari, A. Jaekel, and S. Bandyopadhyay, "Genetic algorithm based approach for extending the lifetime of two-tiered sensor networks," in *Proc. 2nd Int. Symposium on Wireless Pervasive Computing ISWPC*, A. Bari, Ed., 2007.

[78] S. Hussain, A. W. Matin, and O. Islam, "Genetic algorithm for energy efficient clusters in wireless sensor networks," in *Proc. 4th Int. Conf. on Information Technology ITNG*, A. W. Matin, Ed., 2007, pp. 147–154.

[79] F. Xue, A. Sanderson, and R. Graves, "Multi-objective routing in wireless sensor networks with a differential evolution algorithm," in *Proc. IEEE Int. Conf. on Networking, Sensing and Control ICNSC*, A. Sanderson, Ed., 2006, pp. 880–885.

[80] S. Guru, S. Halgamuge, and S. Fernando, "Particle swarm optimisers for cluster formation in wireless sensor networks," in *Proc. Int. Conf. on Intelligent Sensors, Sensor Networks and Information Processing*, S. Halgamuge, Ed., 2005, pp. 319–324.

[81] P. Arabshahi, A. Gray, I. Kassabalidis, M. A. El-Sharkawi, R. J. Marks, A. Das, and S. Narayanan, "Adaptive routing in wireless communication networks using swarm intelligence," in *Proc. the 9th AIAA Int. Communications Satellite Systems Conf.*, Toulouse, France, 2001.

[82] K. M. Sim and W. H. Sun, "Ant colony optimization for routing and load-balancing: Survey and new directions," *IEEE Trans. Syst., Man, Cybern. A*, vol. 33, no. 5, pp. 560–572, 2003.

[83] S. S. Iyengar, H.-C. Wu, N. Balakrishnan, and S. Y. Chang, "Biologically inspired cooperative routing for wireless mobile sensor networks," *IEEE Systems Journal*, vol. 1, no. 1, pp. 29–37, 2007.

[84] S. Bashyal and G. K. Venayagamoorthy, "Collaborative routing algorithm for wireless sensor network longevity," in *Proc. 3rd Int. Conf. on Intelligent Sensors, Sensor Networks and Information Processing ISSNIP*, Dec. 2007.

[85] R. Muraleedharan and L. A. Osadciw, "A predictive sensor network using ant system," in *Proc. Int. Society For Optical Engineering Symposium*, R. M. Rao, S. A. Dianat, and M. D. Zoltowski, Eds., vol. 5440, August 2004, pp. 181–192.

[86] A. Förster and A. L. Murphy, "CLIQUE: Role-Free Clustering with Q-Learning for Wireless Sensor Networks," in *Proc. 29th Int. Conf. on Distributed Computing Systems (ICDCS)*, Montreal, Canada, 2009.

[87] Y. Zhang and M. P. J. Fromherz, "A robust and efficient flooding-based routing for wireless sensor networks," *Journal of Interconnection Networks*, vol. 7, no. 4, pp. 549–568, 2006.

[88] P. Wang and T. Wang, "Adaptive routing for sensor networks using reinforcement learning," in *Proc. 6th IEEE Int. Conf. on Computer and Information Technology (CIT)*. Washington, DC, USA: IEEE Computer Society, 2006.

[89] B. Yu, P. Scerri, K. Sycara, Y. Xu, and M. Lewis, "Scalable and reliable data delivery in mobile ad hoc sensor networks," in *Proc. 4th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, 2006.

[90] D. Shaoqiang, P. Agrawal, and K. Sivalingam, "Reinforcement learning based geographic routing protocol for UWB wireless sensor network," in *Proc. IEEE Global Telecommunications Conf. (GLOBECOM)*, November 2007, pp. 652–656.

[91] O. Islam and S. Hussain, "An intelligent multi-hop routing for wireless sensor networks," in *Proc. WI-IAT Workshops Web Intelligence and Int. Agent Technology Workshops*, Dec. 2006, pp. 239–242.

[92] K. Dasgupta, K. Kalpakis, and P. Namjoshi, "An efficient clustering-based heuristic for data gathering and aggregation in sensor networks," in *Proc. IEEE Wireless Communications and Networking WCNC*, vol. 3, 16–20 March 2003, pp. 1948–1953.

[93] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *System Sciences, 2000. Proc. 33rd Annual Hawaii Int. Conf. on*, Jan. 2000.

[94] G. Gupta and M. Younis, "Load-balanced clustering of wireless sensor networks," in *Proc. IEEE Int. Conf. on Communications ICC '03*, vol. 3, 2003, pp. 1848–1852 vol.3.

[95] G. Di Caro, F. Ducatelle, and L. Gambardella, "AntHocNet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks," *European Trans. on Telecommunications*, vol. 16, pp. 443–455, 2005.

[96] S. Hao and T. Wang, "Sensor networks routing via bayesian exploration," in *Proc. 31st IEEE Conf. on Local Computer Networks*, 2006, pp. 954–955.

[97] Y. J. Shen and M. S. Wang, "Broadcast scheduling in wireless sensor networks using fuzzy hopfield neural network," *Expert Syst. Appl.*, vol. 34, no. 2, pp. 900–907, 2008.

[98] Q. Tang, N. Tummala, S. Gupta, and L. Schwiebert, "Communication scheduling to minimize thermal effects of implanted biosensor networks in homogeneous tissue," *IEEE Trans. Biomed. Eng.*, vol. 52, no. 7, pp. 1285–1294, 2005.

[99] Z. Shi, Z. Zhe, L. Qian-nan, and C. Jian, "Dynamic alliance based on genetic algorithms in wireless sensor networks," in *Proc. Int. Conf. on Wireless Communications, Networking and Mobile Computing WiCOM*, 22–24 Sept. 2006, pp. 1–4.

[100] M. H. Jin, W. Z. Liu, D. Hsu, and C. Y. Kao, "Compact genetic algorithm for performance improvement in hierarchical sensor networks management," in *Proc. 8th Int. Symposium on Parallel Architectures, Algorithms and Networks ISPAN*, 7–9 Dec. 2005.

[101] W. Xue and Z. Chi, "An immune algorithm based node scheduling scheme of minimum power consumption and no collision for wireless sensor networks," in *Proc. NPC Workshops Network and Parallel Computing Workshops IFIP Int. Conf. on*, 18–21 Sept. 2007, pp. 630–635.

[102] C. Pandana and K. J. R. Liu, "Near-optimal reinforcement learning framework for energy-aware sensor communications," *IEEE J. Select. Areas Commun.*, vol. 23, no. 4, pp. 788–797, 2005.

[103] Z. Liu and I. Elahanany, "RL-MAC: A reinforcement learning based MAC protocol for wireless sensor networks," *Int. Journal on Sensor Networks*, vol. 1, no. 3/4, pp. 117–124, 2006.

[104] J. Mao, Z. Wu, and X. Wu, "A TDMA scheduling scheme for many-to-one communications in wireless sensor networks," *Comput. Commun.*, vol. 30, no. 4, pp. 863–872, 2007.

[105] H. Shi and L. Wang, "Broadcast scheduling in wireless multihop networks using a neural-network-based hybrid algorithm," *Neural Netw.*, vol. 18, no. 5-6, pp. 765–771, 2005.

[106] S. Salcedo-Sanz, C. Bousono-Calzon, and A. Figueiras-Vidal, "A mixed neural-genetic algorithm for the broadcast scheduling problem," *IEEE Trans. Wireless Commun.*, vol. 2, no. 2, pp. 277–283, 2003.

[107] G. Wang and N. Ansari, "Optimal broadcast scheduling in packet radio networks using mean field annealing," *IEEE J. Select. Areas Commun.*, vol. 15, no. 2, pp. 250–260, 1997.

[108] J. Ignizio, *Introduction to Linear Goal Programming*. Sage Publications, 1986.

[109] M.-H. Jin, D. F. H. Yu-Cheng Huang, C.-Y. Kao, Y.-R. Wu, and C.-K. Lee, "On active interval scheduling in static sensor networks," in *Proc. IASTED Int. Conf. on Communication Systems and Applications*, July 2004, pp. 126–131.

[110] B. Lazzerini, F. Marcelloni, M. Vecchio, S. Croce, and E. Monaldi, "A fuzzy approach to data aggregation to reduce power consumption in wireless sensor networks," in *Proc. Annual meeting of the North American Fuzzy Information Processing Society NAFIPS*, 3–6 June 2006, pp. 436–441.

[111] I. V. Maslov and I. Gertner, "Multi-sensor fusion: An evolutionary algorithm approach," *Inf. Fusion*, vol. 7, no. 3, pp. 304–330, 2006.

[112] Q. Wu, N. Rao, J. Barhen, S. Iyengar, V. Vaishnavi, H. Qi, and K. Chakrabarty, "On computing mobile agent routes for data fusion in distributed sensor networks," *IEEE Trans. Knowledge Data Eng.*, vol. 16, no. 6, pp. 740–753, June 2004.

[113] S. Aldosari and J. Moura, "Fusion in sensor networks with communication constraints," in *Proc. 3rd Int. Symposium on Information Processing in Sensor Networks IPSN*, 2004, pp. 108–115.

[114] N. Gnanapandithan and B. Natarajan, "Parallel genetic algorithm based optimal fusion in sensor networks," in *Proc. 3rd IEEE Consumer Communications and Networking Conf. CCNC*, vol. 2, 2006, pp. 763–767.

[115] S. A. Munir, Y. W. Bin, R. Biao, and M. Jian, "Fuzzy logic based congestion estimation for QoS in wireless sensor network," in *Proc. IEEE Wireless Communications and Networking Conf. WCNC 2007*, March 2007, pp. 4336–4341.

[116] L. Khoukhi and S. Cherkaoui, "FuzzyCCG: A fuzzy logic QoS approach for congestiosn control in wireless ad hoc networks," in *Proc. 1st ACM int. workshop on Quality of service & security in wireless and mobile networks, Q2SWinet*. New York, NY, USA: ACM, 2005, pp. 105–111.

[117] L. Khoukhi and S. Cherkaoui, "Experimenting with fuzzy logic for QoS management in mobile ad hoc networks," *Int. Journal of Computer Science and Network Security*, vol. 8, no. 8, pp. 372–386, August 2008.

[118] G. K. Venayagamoorthy, R. G. Harley, and D. C. Wunsch, "Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator," *IEEE Trans. Neural Networks*, vol. 13, no. 3, pp. 764–773, May 2002.

Raghavendra V. Kulkarni (M'97–SM'05) received his B.E. degree in electronics & communication engineering from Karnatak University, India, in 1987, and M. Tech. degree in electronics engineering from the Institute of Technology, Banaras Hindu University, India, in 1994. Prior to 2006, he served Gogte Institute of Technology, Belgaum, India, as an assistant professor. He is currently working towards his PhD degree in electrical engineering in the Missouri University of Science and Technology (Missouri S&T), Rolla, USA. His research interests include the development of wireless sensor network applications using computational intelligence tools.

Mr. Kulkarni was the registration and publications chair of the 2008 IEEE Swarm Intelligence Symposium (SIS'08). He is a life member of the Indian Society for Technical Education (ISTE), and a member of the IEEE Computational Intelligence Society and the International Neural Network Society (INNS).

Anna Förster (S'08–M'09) received her PhD from the University of Lugano, Switzerland and her Masters degree from the Free University in Berlin, Germany. Currently she is a postdoctoral researcher at the University of Lugano, Switzerland. She has worked on various topics in Robotics and Artificial Intelligence, before concentrating on communication, implementation and deployment issues of wireless sensor networks. Her current research interests include intelligent communication, evaluation and deployment optimization in wireless sensor networks, vehicular networking, delay-tolerant networking, and the interplanetary internet.

Dr. Förster is a member of Association for Computing Machinery (ACM), IEEE Communication Society and the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering (ICST). She is serving as a steering committee member for the international OMNeT++ workshop and as a technical programm committee member and a reviewer for numerous top-quality conferences and journals in wireless networking and simulation.

Ganesh Kumar Venayagamoorthy (S'91–M'97–SM'02) received his Ph.D. degree in electrical engineering from the University of KwaZulu Natal, Durban, South Africa, in Feb. 2002. Currently, he is an Associate Professor of Electrical and Computer Engineering, and the Director of the Real-Time Power and Intelligent Systems (RTPIS) Laboratory at Missouri University of Science and Technology (Missouri S&T). He was a Visiting Researcher with ABB Corporate Research, Sweden, in 2007. His research interests are in the development and applications of advanced computational algorithms for real-world applications, including power systems stability and control, smart grid, sensor networks and signal processing. He has published 2 edited books, 5 book chapters, over 70 refereed journals papers and 250 refereed conference proceeding papers. He has been involved in approximately US$ 7 Million of competitive research funding.

Dr. Venayagamoorthy is a recipient of several awards, including a 2007 US Office of Naval Research Young Investigator Program Award, a 2004 US National Science Foundation CAREER Award, the 2008 IEEE St. Louis Section Outstanding Educator Award, the 2006 IEEE Power Engineering Society Walter Fee Outstanding Young Engineer Award, the 2005 IEEE Industry Applications Society (IAS) Outstanding Young Member Award, the 2003 International Neural Network Society (INNS) Young Investigator Award, and Missouri S&T 2008, 2007 and 2006 Faculty Excellence Awards, 2006 Teaching Excellence Award and 2007 Teaching Commendation Award.

Dr. Venayagamoorthy has been involved in the leadership and organization of many conferences including the General Chair of the 2008 IEEE Swarm Intelligence Symposium (St. Louis, USA) and the Program Chair of 2009 International Joint Conference on Neural Networks (Atlanta, USA). He is currently the Chair of the IEEE Power and Energy Society (PES) Working Group on Intelligent Control Systems, the Chair of IEEE Computational Intelligence Society (CIS) Task Force on Power Systems Applications, the Vice-Chair of the IEEE PES Intelligent Systems Subcommittee, and the Chair of IEEE CIS and IEEE Industry Applications Society St. Louis Chapters. He is a Fellow of the Institution of Engineering and Technology (IET), UK and the South African Institute of Electrical Engineers (SAIEE), a Senior Member of the INNS, and a Member of the INNS Board of Governors.