A Neural Network Model to Minimise the Connected Dominating Set for Self-Configuration of Wireless Sensor Networks

Hongmei He, Member, IEEE, Zhenhuan Zhu and Erkki Mäkinen

Abstract—A wireless ad hoc sensor network consists of a number of sensors spreading across a geographical area. The performance of the network suffers as the number of nodes grows, and a large sensor network quickly becomes difficult to manage. Thus it is essential that the network is able to self-organize. Clustering is an efficient approach to simplify the network structure and to alleviate the scalability problem.

One method to create clusters is to use weakly connected dominating sets (WCDS). Finding the minimum WCDS in an arbitrary graph is an NP-complete problem. We propose a neural network model to find the minimum WCDS in a wireless sensor network. We present a directed convergence algorithm. The new algorithm outperforms the normal convergence algorithm both in efficiency and in the quality of solutions. Moreover, it is shown that the neural network is robust.

We investigate the scalability of the neural network model by testing it on a range of sized graphs and on a range of transmission radiuses. Compared with Guha and Khuller’s centralised algorithm, the proposed neural network with directed convergency achieves better results when the transmission radius is short, and equal performance when the transmission radius becomes larger. The parallel version of the neural network model takes time $O(dn)$, where $d$ is the maximal degree in the graph corresponding to the sensor network, while the centralised algorithm takes $O(n^2)$.

We also investigate the effect of the transmission radius on the size of WCDS. The results show that it is important to select a suitable transmission radius to make the network stable and to extend the lifespan of the network. The proposed model can be used on sink nodes in sensor networks, so that a sink node can inform the nodes to be a coordinator (clusterhead) in the WCDS obtained by the algorithm. Thus the message overhead is $O(\mathcal{M})$, where $\mathcal{M}$ is the size of the WCDS.

Index Terms—Neural networks, Connected Dominating Set, Directed Convergence, Transmission Radius, Wireless sensor networks

I. INTRODUCTION

Typically, there is a large number of (mostly stationary) sensors in a sensor network. Networks of 10,000 or even 100,000 nodes are envisioned, so scalability is a major issue. Moreover, nodes may fail (either from lack of energy or from physical destruction), individual nodes may become disconnected from the rest of the network, and new nodes may join the network. Therefore, the network must be able to periodically reconfigure itself so that it can continue to function. The multi-cluster mechanism enables the formation of a scalable network topology by allowing interconnected clusters in the network. Each cluster is independent of others, and is assigned a distinct channel. Certain nodes must be members of multiple clusters to allow network connectivity.

On the other hand, due to limited radio range and battery power, most nodes cannot communicate directly with a base station, but rather rely on their peers to forward messages to and from base stations. Likewise, in mobile ad hoc networks (MANETs), the routing of messages is performed by ordinary nodes. In fact, a MANET typically has no network infrastructure, and therefore all routing and network management functions must be performed by ordinary nodes. The key to scalability and efficiency in traditional computer networks is the organization of the network infrastructure into a hierarchical structure. However, due to the lack of a network infrastructure, sensor networks and MANETs are inherently flat. In order to achieve scalability and efficiency, some new algorithms have emerged to organise ordinary nodes into a hierarchy. Thus, given a large number of nodes and their potential placement in hostile locations, it is essential that the network is able to self-organise by selecting the virtual backbone of the network.

The construction of this infrastructure in wireless networks is an application of connected dominating sets (CDSs) in a graph. The utility of CDSs in wireless ad hoc networks has been demonstrated in protocols that perform a wide range of communication functions [4]. CDSs have formed an underlying architecture used by protocols including media access coordination [3], [19], [34]; unicast [43], multicast/broadcast [28], [35], [41], [44]; and location-based routing [13], [14]; energy conservation [8], [16], [33], [42], [45]; and topology control [15], [16]. CDS can also be used to facilitate resource discovery in MANET [24], [27].

There are many algorithms for constructing CDSs in a graph. They can be divided into two categories, centralised [12], [20], [30], [32] and distributed [1], [5], [6], [10], [11], [13], [38], [39]. WCDS construction plays an important role not only in clustering algorithms, but also in a number of CDS construction algorithms, which begin with the selection of a
A Hopfield network is a fully connected recurrent single layer and unsupervised network. Hopfield and Tank [26] were the first to use a neural network model for solving optimisation problems. In this paper, we propose an efficient neural network model to minimise the weakly connected dominating sets of a connected graph which models a wireless sensor network.

II. Preliminaries

A sensor network can be represented by a graph \( G(V, E) \) comprised of a set of time-varying vertices \( V \) and edges \( E \). For each pair of vertices \( u, v \in V \), \( (u, v) \in E \) if and only if the nodes \( u \) and \( v \) are within communication range at time \( t \). However, due to the relative stability of sensor networks over some period of time, we can simply use \( G(V', E') \) to represent the sensor network.

Given omni-directional antennae, the communication range of a node in a wireless network is typically modeled as a disk centered at the node with radius equal to the transmission range of the radio. Consequently, when transmission range is fixed for all nodes, the network has the property of a unit-disk graph (UDG), where an edge exists if and only if two nodes have inter-nodal distance less than or equal to 1 unit (the fixed communication range).

In a graph, a vertex is said to dominate each of its adjacent vertices (Fig. 1 (a)). A dominating set (DS) of a graph is a set of vertices which together dominate the rest of the vertices (Fig. 1 (b)). In other words, a set \( S \) is dominating if each node in a graph is either in \( S \) or adjacent to at least one of the nodes in \( S \). A graph can have numerous different dominating sets of varying sizes. Often it is interesting to know the smallest possible dominating set for a graph. This is called a minimal dominating set, and the number of vertices in such a set is called the dominating number.

The first stage is to construct a DS through MIS formation, which initially elects a leader among non-localized computation. Chen and Liestman [9] also proposed a zonal algorithm (referred to as ZONAL). In this algorithm, the graph is divided into regions, for each of which, a WCDS is constructed, and an adjustment along the borders of regions is done to produce a WCDS for the whole graph. When the graph is small, ZONAL constructs a relatively large WCDS, whereas for larger graphs the size of WCDS constructed are relatively small. It may lead to a large message overhead [22].

Guha and Khuller [20] presented two centralized greedy heuristic algorithms for connected dominating set formation. In the first algorithm, the CDS is grown from one node outward. In the second algorithm, a WCDS is constructed (we refer to this phase as CENTRAL), and then intermediate nodes are selected to create a CDS. CENTRAL begins by marking all vertices white, and a piece is defined to be either a connected black component, or a white node. Iteratively it selects a node that can decrease the maximum number of pieces. Once a node is selected, it is marked black and its white neighbours are marked gray. CENTRAL terminates when there are no white nodes left. The distributed implementations of both algorithms were provided by Das and Bharghavan [13].

Chen and Liestman [10] proposed a series of approximate algorithms for computing a small WCDS to cluster mobile ad hoc networks. One of the proposed distributed approximation algorithms is inspired by CENTRAL. To solve the problem of non-localized computation, Chen and Liestman [9] also proposed a zonal algorithm (referred to as ZONAL). In this algorithm, the graph is divided into regions, for each of which, a WCDS is constructed, and an adjustment along the borders of regions is done to produce a WCDS for the whole graph. When the graph is small, ZONAL constructs a relatively large WCDS, whereas for larger graphs the size of WCDS constructed are relatively small. It may lead to a large message overhead [22].

Alzoubi et al. [2] presented two distributed algorithms for constructing a WCDS for wireless ad hoc networks in linear time. Both algorithms are used to obtain sparse spanners based on the construction of a maximal independent set (MIS) [1]. The first algorithm initially elects a leader \( r \) among the nodes, which becomes the root of a spanning tree. The second algorithm first constructs a MIS and then modifies it by selecting one intermediate node between each pair of dominators separated by exactly three hops. Following Han and Jia [22], we refer to the second algorithm as AWF.

Recently, Han and Jia [22] proposed a distributed algorithm (referred to as AREA). This algorithm includes two stages. The first stage is to construct a DS through MIS formation,
for which, the wireless nodes are divided into several areas naturally and dominating nodes in each area form a WCDS of that area automatically. The second stage is to add additional nodes to the DS to make it weakly-connected by adjusting the area borders to make areas weakly-connected. In this algorithm, min-ID and max-degree can be used as criteria for choosing clusterheads. We refer to these variants as AREA-ID and AREA-degree, respectively.

IV. A HOPFIELD NEURAL NETWORK MODEL

For spatial data analysis, an artificial neural network can be used to model a graph by modeling either a node [21] or an edge [7], [23] in the graph as a neuron, depending on the problem to be solved. Here, we map a neuron to a node in a connected graph $G$. For a connected graph $G$ with $n$ nodes, there should be $n$ neurons in the neural network. In a model of binary neurons [29], the fired neurons represent the dominating vertices, while the unfired neurons indicate the nodes that do not belong to the dominating set.

Suppose we have a graph $G=(V, E)$ with edges $\{e(i,j) \in E, i, j \in V\}$, and $n$ vertices $V \rightarrow (0, ..., n-1)$. Consider a neural network consisting of $n$ neurons $v_0, ..., v_{n-1}$. If a neuron $v_i$ is fired, then the neurons mapped to the adjacent vertices of $v_i$ should not be fired. If a neuron $v_i$ is not fired, then among those neurons mapped to adjacent vertices of $v_i$, only one should be fired. Thus we give the energy function as

$$E = A \sum_{i=0}^{n-1} \sum_{j=0, j \neq i}^{n-1} e(i,j)v_iv_j + B \sum_{i=0}^{n-1} (1 - \sum_{j=0, j \neq i}^{n-1} e(i,j)v_j)^2 v_i.$$  

(1)

The first term of Equation (1) will be zero when all adjacent neurons of each fired neuron are unfired, and the second term will be zero, when only one among all adjacent neurons of each unfired neuron is fired. Ideally, when both the terms are zero, namely, $E$ arrives at a zero state, the model outputs the optimal solution for the graph that is mapped to the wireless sensor network. However, the energy function is not linear, and the two terms will interfere each other. Therefore, the neural network will arrive at a stable local optimal state. For a binary neuron, it holds $\bar{v_i} = 1-v_i$. Thus, Equation (1) can be written as

$$E = A \sum_{i=0}^{n-1} \sum_{j=0, j \neq i}^{n-1} e(i,j)v_iv_j + B \sum_{i=0}^{n-1} (1 - \sum_{j=0, j \neq i}^{n-1} e(i,j)v_j)^2 (1-v_i).$$  

(2)

Hirsch [25] viewed a neural network as a nonlinear dynamic system called Neurodynamics, which presents a conceptual and eclectic methodological approach for understanding neural network activity. Given a dynamic system with $n$ state variables $v_0, v_1, ..., v_{n-1}$, the network motion equation is $du_i/dt = -\partial E/\partial v_i$, where $u_i$ and $v_i$ are the input and output of the $i$-th neuron. Therefore we can write the motion equation as

$$\frac{du_i}{dt} = -A \sum_{j=0, j \neq i}^{n-1} e(i,j)v_j + B(1 - \sum_{j=0, j \neq i}^{n-1} e(i,j)v_j)^2.$$  

(3)

V. SIMULATION

A. Random convergence

Takefuji [36], [37] proved that the state of the binary model always converges to a local minimum. For a normal Hopfield neural network,

$$u_i = \sum_{j=1, j \neq i} w_{ij}v_j - \theta_i,$$

where $w_{ij}$ is the synaptic weight from neuron $j$ to neuron $i$, and $\theta_i$ is the threshold. The convergence can be formalised as follows:

Theorem 5.1: [40] Starting from any initial configuration, any symmetric neural network with energy function $E$ computing in a sequential mode will achieve a stable state after at most $O(p)$ computational cycles, where $p = \frac{1}{A} \sum_{i=1}^{n} |w_{ij}| + \sum_{i=1}^{n} |\theta_i|$. Moreover, this stable state represents a local minimum of $E$.

According to the motion equation (3), there are two kinds of forces in the neural network, excitatory and inhibitory. If a neuron $v_i$ is fired, the number of all fired adjacent vertices of the vertex mapped to the fired neurons should be zero. Otherwise, the first term in the motion function will be an inhibitory force to make the neuron unfired. When a neuron $v_i$ is not fired, the number of fired neurons in all adjacent vertices of $v_i$ should be one. Conversely, if no adjacent vertex is fired, then the second term will be an excitatory force to encourage the unfired neuron $v_i$ to be fired. The second term is to avoid the case that all neurons are not fired.

We assume $\Delta t = 1$. According to the motion function, we can calculate the input of each neurons by updating $u_i$ with $\Delta u_i$. Algorithm 1 shows the convergence procedure with a sequential model, in which, the activation function has the form:

$$v_i = \begin{cases} 1 & \text{if } u_i > 0 \\ 0 & \text{otherwise.} \end{cases}$$

(4)

For the parallel model, the running time of Algorithm 1 is $O(d)$, where $d$ is the maximal degree on the vertices. We
can consider $d$ as a constant, and say that the running time of the parallel mode is $O(1)$. From Theorem 5.1, we know that a sequential model always converges to a local optimal state. Therefore, we can simply apply a hill climbing approach to find out the best solution from “the local minimum” (see Algorithm 2).

Algorithm 2 hillClimbing

1: $optE = MAX\_VALUE$;
2: $k = 0$;
3: while ($k < MAX\_ITERATION$) and $optE \neq 0$ do
4:  $currS = InitNet();$
5:  $currE = calculateEnergy(currS);$ 
6:  $optE = currE;$
7:  if ($optE > currE$) then
8:      saveState(currS, optState);
9:      $optE = currE;$
10:  end if
11:  $k = k + 1$;
12: end while

B. Directed convergence

According to dynamics, an object will move along the direction of the composition of forces that act on the object. In the motion functions, each term can be viewed as a force that acts on a neuron. Only when the both terms are zero, the motion function will be zero. However, the dynamic system always gets a composition of the forces, which makes the system overcompensated or uncompensated, and it always converges along a random direction. Can we find an effective path to let the network converge? Usually, it is not easy. However, for the problem of minimising the size of WCDS, the answer is affirmative.

Intuitively, the size of WCDS can be reduced if the dominating set contains high degree vertices. Thus, we sort the vertices in descending order of their degree. The vertices with larger degree will have a higher priority to be computed.

However, another issue rises. The network will converge to a local optimal state quickly, and it is difficult to jump out of the local optimal state. Therefore, we present a heuristic algorithm to make the network break away from the “deadlock”. Two extreme cases are considered. First, if a vertex is not dominating (the neuron is unfired) and its neighbouring neurons are all unfired, then let the neuron be fired. Secondly, if a neuron is fired and its neighbouring neurons are all fired, then let all neighbours be unfired.

In general, we aim at low energy networks. Sometimes, the network falls into a state with fewer dominating vertices, but with more two-hop non-dominating vertices (i.e., vertices which need two hops to transfer information to a dominating vertex). We define a performance measure $p = k_1 \times k_2 \times E$, where $k_1$ is the number of vertices in the dominating set, $k_2$ is the number of vertices that can be visited in two hops, and $E$ is the energy of a state. If the current state has a higher energy, but the value $p$ of the current state is less than that in the best state so far found, we record the solution as a new approximation of the optimal solution. This procedure will avoid missing some solutions with a little more dominating vertices but fewer two-hop non-dominating vertices.

The following threshold is used in activation function:

$$v_i = \begin{cases} 1 & \text{if } u_i > \theta \\ 0 & \text{otherwise} \end{cases}$$

where $\theta$ is a small positive real number. Actually, the use of the threshold is equivalent with adding the term, $E_3 = \theta \sum_{i=1}^{n-1} v_i$, to the energy function. It is an inhibitory force to make fewer neurons fired. Thus,

$$\frac{du_i}{dt} = -A \sum_{j=0,j\neq i}^{n-1} e(i,j)v_j + B(1 - \sum_{j=0,j\neq i}^{n-1} e(i,j)v_j)^2 - \theta.$$

If $u_p$ and $u$ are the input signals obtained by the previous and new energy functions, respectively, we have $u = u_p - \theta$. If we use the previous activation function, then $u = u_p - \theta > 0$, i.e., $u_p > \theta$.

Therefore, we have an improved hill climbing algorithm (see Algorithm 3).

Algorithm 3 Improved hill climbing

1: $currS = InitNet();$
2: $currE = calculateEnergy(currS);$ 
3: saveState(optS, currS);
4: $optE = currE;$
5: $k_1 = getFiredNeuronNo(optS);$ 
6: $k_2 = count2HopNeurons(optS);$ 
7: $old\_ev = k_1 \times k_2 \times optE;$
8: while ($k < MAX\_ITERATION$) do
9:  $currS = directedConverge();$
10:  $currE = calculateEnergy(currS);$ 
11:  $k_1 = getFiredNeuronNo(currS);$ 
12:  $k_2 = count2HopNeurons(currS);$ 
13:  $new\_ev = k_1 \times k_2 \times currE;$
14:  if ($optE > currE \text{ or } old\_ev > new\_ev$) then
15:      saveState(optS, currS); 
16:      $optE = currE; \text{ old\_ev = new\_ev;}$
17:  end if
18:  $currS = Heuristic();$
19:  /*to make the network to jump out of a local minimum*/
20:  $currE = calculateEnergy(currS);$ 
21:  $k_1 = getFiredNeuronNo(currS);$ 
22:  $k_2 = count2HopNeurons(currS);$ 
23:  $new\_ev = k_1 \times k_2 \times currE;$
24:  if ($optE > currE \text{ or } old\_ev > new\_ev$) then
25:      saveState(optS, currS); 
26:      $optE = currE; \text{ old\_ev = new\_ev;}$
27:  end if
28:  $k = k + 1$;
29: end while
VI. Experiments and Evaluation

A sequential simulator of the convergence algorithm was developed for the tests. In the sequential simulator the output value of each neuron is individually computed as long as the input of the neuron is evaluated in the loop of the motion equation. The algorithm was implemented in C++ and ran on a laptop with CPU 1.7GHZ. The test graphs include complete graphs, meshes, and UDGs.

A. Test with the random convergence algorithm

1) Test on a complete graph: Finding suitable values for parameters \( A \) and \( B \) is a “trial-and-error” process [31]. Ideally, to improve the solution quality, all parameters should be adjusted to fit each problem instance (different kinds of graphs). This requires extensive experimentation before a good set of values can be found.

In order to verify the convergence of the neural network, we first tested the random convergence algorithm on a complete graph. In a complete graph each pair of vertices is adjacent, and thus, any vertex alone dominates the graph. We first trained the neural network on the complete graph \( K_{15} \) to get suitable parameters \( A \) and \( B \). The initial state of neuron network is decided by the initial input of +0.5 or -0.5 with probability \( f = 0.5 \) each. We fixed the parameter \( A = 0.5 \), ran the model on \( K_{15} \) for a range of \( B \) from 0.01 to 0.12 with step 0.02. Fig. 2 shows the energy of the neural network with the same initial state converges to 0, when \( B < 0.12 \). The experimental results indicate that the model achieves the optimal solution in the first traversal of any complete graph for the parameters above. Moreover, the algorithm runs extremely fast: the time of each convergence procedure is less than 1ms.

We also examined the performance of the model when parameter \( B \) is fixed and parameter \( A \) varies in \((0,2)\). Similarly, the initial state of each neuron in the network is +0.5 or -0.5 with equal probability. The model was tested on the complete graph \( K_{15} \) with \( B = 0.01, B = 0.06 \) and \( B = 0.12 \), respectively. There are three possible periods, volatilising \((p_1)\), uncertain \((p_2)\) and certain \((p_3)\). The volatilising period is defined as the interval of parameter \( A \), in which, the neural network can not converge to a local optimal state, and all neurons will finally be fired. The uncertain period is defined as the interval of \( A \), in which, the neural network can converge to local states, but the number of fired neurons varies with parameter \( A \). The certain period is defined as the interval of \( A \), in which, the neural network converge to the optimal state, and only one neuron is fired. Table I shows the periods of parameter \( A \) dependent to parameter \( B \).

In the case of the mesh graph \( P_{10} \times P_{10} \), the stable state was found usually after two or three traversals of all the neurons. Fig. 3 shows the curves of energy for ten hill climbing calls. After the first traversal of all neurons, energy has become very small. So, only the changes of energy in the first traversal for each hill climbing are presented here. The time of each convergence process was at most 10 ms. After ten hill climblings a nearly optimal solution was found. Fig. 4 shows the best solution for the mesh graph \( P_{10} \times P_{10} \) in 20 tests, where the \( i^{th} \) row and the \( j^{th} \) column vertex is denoted as \((i,j)\). There are 28 fired neurons (dominating vertices) out of the total 100 neurons, and the remains are one-hop or two-hop non-dominating vertices. There are 12 two-hop non-dominating vertices. For example, a message from vertex \((8,3)\) to vertex \((9,4)\) (a two-hop vertex) must pass through \((9,3)\) or \((8,4)\). Experimental results show that some solutions have a smaller dominating set, but the number of two-hop vertices

![Fig. 2. Energy convergence curves for different parameters](image1)

![Fig. 3. Energy convergence curves for 10 random convergence hill climblings](image2)

![Fig. 4. Energy convergence curves for 10 random convergence hill climblings](image3)
reaches 25%. In general, for most solutions the number of two-hop vertices is between 14 and 20, and the size of the dominating set is between 20 and 40.

B. Test with the directed convergence algorithm

The tests show that some two-hop non-dominating vertices are produced in meshes when using the random convergence hill climbing. In the application, this would result in a disconnected network. Hence, there should be as few two-hop vertices as possible. We implemented the improved hill climbing algorithm with directed convergence, heuristic adjustment and threshold compensation. The probability \( f \) that the initial state of a neuron is 1, is set to be 0.6. The neural network was simply trained to decide the parameters \( A \), \( B \) and \( \theta \) in turn. First, fixing the parameter \( A \) to be 0.28, and \( \theta \) to be 0, when the parameter \( B \) is at a range from 0.001 to 0.117, the neural network has good performance: the size of dominating set is between 26 and 33, and the number of two-hop non-dominating vertices is less than 5. Secondly, fixing \( A = 0.28, B = 0.011 \), slightly adjust parameter \( \theta \). Finally, the following set of parameters were used: \( A = 0.28, B = 0.011, \theta = 0.03 \), and \( f = 0.6 \). We tested the mesh graph with the improved hill climbing algorithm. The constant \( MAX_{\text{ITERATION}} \) was set to be 5.

Similar as in Section VI-A1, we can validate parameter \( A \) when fixing parameter \( B \). There exists a critical value \( \alpha \) of \( A \), and when \( A \geq \alpha \) at \( B = 0.011 \), the model can arrive at an local optimal state. Experiments show that the neural network can converge to an optimal state when \( B = 0.011 \) and \( A > 0 \), and achieved very good performance. Fig. 5 shows the best solution obtained for \( P_{10} \times P_{10} \) with the improved hill climbing algorithm in 20 tests. The size of the dominating set is 29, and there are no two-hop non-dominating vertices. For the most cases, the size of the dominating set varies from 26 to 35, and the number of two-hop non-dominating vertices is between 0 and 5. Usually, after two or three hill climblings the network arrives at a stable state. Obviously, the improved hill climbing algorithm obtained better solution for the tested mesh.

C. Scalability

Assuming that all sensor nodes are randomly and uniformly distributed in an area \( \mathcal{A} \) of size \( a \times a \) units, and that each node has a fixed transmission range \( r \), the density \( \mu \) of a graph can be calculated as \( \mu(r) = \frac{n\pi r^2}{\mathcal{A}} \), where \( n \) is the number of nodes in the graph. Actually, \( \mu(r) \) presents the average number of nodes within the transmission radius of each node. If the \( n \) nodes of a graph are evenly distributed as 2-dimension array in the area \( \mathcal{A} \), then the distance between two neighbouring nodes will approximately be \( \delta = \sqrt{\frac{\mathcal{A}}{n}} \), and a node \( u \) will be connected with those nodes that are allocated in the range of transmission radius \( r \) (see Figure 6). The distance between node \( u_{ij} \) and node \( u_{kl} \) can be calculated with the following formula:

\[
\delta = \delta \sqrt{(k-i)^2+(l-j)^2}.
\]

1) Fixed transmission radius: The experiments were carried out in the area of size 100 by 100 units, based on fixed transmission radiiuses 15 and 30 units, and the numbers of nodes ranging from 49 to 961, corresponding to the numbers of nodes in each row and column ranging from 7 to 31 with increasing step 1. When the number of nodes in each row is less than 7, the graph is not connected for transmission radius \( r = 15 \), as \( \delta = 100/6 = 16.7 > 15 \). For transmission radius \( r = 30 \), the number of nodes in each row should be not less than 4.
Han and Jia [22] compared their algorithm AREA with ZONAL [9], AWF [1] and CENTRAL [20]. They found that AREA-degree outperforms ZONAL and AWF when there are more than 100 nodes in the tested area. However, AREA-degree is worse than CENTRAL, for all tested graphs with transmission radius \( r = 15 \) or 30. AREA-degree achieves better results than AREA-ID.

Therefore, we compare the results with the ones obtained by CENTRAL (we have implemented the algorithm). Every graph was tested by the neural network with direct convergency (NNDC) for 10 times, and the average size of WCDS obtained was calculated. The standard deviation was calculated by using the non-biased method:

\[
\sqrt{\frac{1}{n-1}\sum(x - \bar{x})^2}.
\]

The robustness of parameters \( A \) and \( B \) was previewed with the same training method as in Section VI-B. When \( B = 0.00035 \) and \( A \geq 0.0046 \), the network converges well for the UDG with \( 20 \times 20 \) vertices. When \( A = 0.001 \) and \( B \in [0.00001, 0.072] \), the network converges well for the same UDG. In the following experiments, the parameters were set as: \( A = 0.07, B = 0.000035, \theta = 0.00001, f = 0.5 \). The \textsc{MAX\_ITERATION} was set to be 3. Figures 7 and 8 show the results by NNDC and CENTRAL for \( r = 15 \) and \( r = 30 \), respectively. The values shown in the figures are averages of 10 runs. Fig. 9 shows the standard deviations of WCDS size for a range of UDGs at two transmission radiuses \( r=15 \) and \( 30 \), respectively.

2) A range of transmission radiuses: Another important experiment is to observe the effect of ranged transmission radiuses on the size of WCDS. The experiments were done based on the fixed numbers of nodes 1024 and 2025 for a range of transmission radiuses from 5 to 45 increasing with step 5. When 1024 nodes are evenly aligned as \( 32 \times 32 \) array in area \( \mathcal{A} = 100 \times 100 \) units, \( \delta = 100/32 = 3.125 < 5 \). Thus the tested graphs for the range of transmission radiuses are connected graphs in the area \( \mathcal{A} \). Figures 10 and 11 show the results under the conditions described above. The values shown in the figures are averages of 10 runs. Fig. 12 shows the standard deviations of WCDS size at the range of transmission radiuses above for the two UDGs with 1024
Fig. 10. The sizes of WCDSs obtained by NNDC and CENTRAL respectively, when the number of nodes is 1024 (=32 × 32).

Fig. 11. The sizes of WCDSs obtained by NNDC and CENTRAL respectively, when the number of nodes is 2025 (=45 × 45).

and 2025 nodes, respectively. It can be seen that the standard deviation fluctuates with the change of the transmission radius, and it is under the value of 1.5 for the UDG with 1024 nodes. For the UDG with 2025 nodes, the standard deviation decreases on the transmission radius.

Fig. 12. The standard deviation of WCDS size at arrange of transmission radiuses from 5 to 45 with step 5 for UDGs with 1024 and 2025 nodes, respectively, produced by NNDC.

D. Evaluation

1) Size of WCDS: Han and Jia’s [22] experimental results show that the number of nodes in the WCDS constructed by AREA is only about 50% of that constructed by AWF when the number of nodes in the network reaches 1000, and CENTRAL can generate WCDS with the smallest size for the tested graphs. CENTRAL can produce at most |OPT| connected black components, where |OPT| stands for the size of the optimal connected dominating set [20]. From Figures 7 and 8, it can be seen that the neural network has obtained slightly better performance than CENTRAL. Table II lists the statistic values comparing the sizes of dominating sets obtained by NNDC and CENTRAL for all testes UDGs. The size of the dominating set obtained by NNDC is denoted by M_N, and the size of the dominating set obtained by CENTRAL is denoted by M_C.

<table>
<thead>
<tr>
<th>radius</th>
<th>M_N &gt; M_C</th>
<th>M_N = M_C</th>
<th>M_N &lt; M_C</th>
</tr>
</thead>
<tbody>
<tr>
<td>r = 15</td>
<td>5</td>
<td>6</td>
<td>14</td>
</tr>
<tr>
<td>r = 30</td>
<td>9</td>
<td>6</td>
<td>10</td>
</tr>
</tbody>
</table>

2) Increasing density of graphs on a fixed transmission radius: When the transmission radius r and the area A are fixed, the number (D) of disks in the area will be fixed, and can be calculated by $\lceil \frac{A}{\pi r^2} \rceil$. For a connected graph that is evenly distributed in an area, if each disk is aligned only one dominating node, there are totally at most D dominating nodes. For $A = 100 \times 100$, $r = 15$, $D = \lceil \frac{10000}{15^2 \pi} \rceil = 15$, and for $r = 30$, $D = 4$. The experimental results show the sizes of WCDSs produced by NNDC and CENTRAL are less than 4D.

From Fig. 9, it can be seen that the deviation of the results is less than 1.5 for all tested graphs. The standard deviations for some small UDGs are zero, and most nonzero standard deviations occur for those UDGs, of which, the node number per row is between 25 and 31 (i.e., those UDGs with larger edge density). This could be improved by adjusting the parameters of the model.

The results also indicate that the size of WCDS is not related to the size of the graph when the transmission radius is fixed. Although the density of the graph in the area is increased, the number of disks in the area is not changed. Therefore, for real applications, in order to keep the stability of a wireless sensor network, for a fixed period T, the clusterheads should not be changed, whenever a non-clusterhead leaves or is added, unless a clusterhead leaves or its power decreases so that the transmission radius is reduced.

3) Increasing transmission radiuses on a fixed size of graph: Figures 10 and 11 show that NNDC achieves better results than CENTRAL when transmission radius is small. With the increasing of transmission radius, NNDC has the same performance as CENTRAL. It can be seen that the size of
WCDS decreases as the transmission radius increases for a fixed size of graph. The transmission radius \( r \) is a very important parameter for network lifespan. If \( r \) is too large, the number of clusterheads will be small, but each clusterhead will stand for too many neighbours and the power consumption of RF transfer will be large. Thus the effective \( T \) will be small, and consequently the stability and lifespan of a wireless network will be reduced. On the other hand, if transmission radius is too small, the number of clusterheads will be large. This will increase the data flow in the network, and in order to guarantee the connectivity of the network, the number of nodes should be increased. Therefore, it is necessary to find a suitable \( r \) to make the network have a good stability and a long lifespan.

Fig. 12 shows the standard deviations for most tests are less than 2. Only when \( N = 2025 \) and \( r = 5 \), the standard deviation arrives 4.5. The interesting thing is that the standard deviation decreases when radius increases, and it is more obvious for \( N = 2025 \). This is because, when radius is small, for a large evenly distributed UDG, the number of disks increases, and the number of possible dominating sets increase.

4) Message overhead: Alzoubi et al. claimed that their first algorithm has an approximation ratio of 5, and requires \( O(n \log n) \) messages [2], and the second algorithm (AWF) has a larger approximation ratio, and requires only \( O(n) \) messages [1]. The algorithm AREA [22] has message complexity \( O(n) \). Actually, for a distributed algorithm, each node in a wireless network needs to know the information of its neighbours, from a node outward, each node must communicate with its neighbours, therefore message overhead of the network at least \( O(d \cdot n) \), where \( d \) is the degree of a node. Assume the sink node knows the information of network topology, when the network is initialised. If the algorithm runs in sink node, the sink node can inform clusterheads corresponding to the nodes in WCDS produced by the algorithm. Therefore, the message overhead is \( O(\mathcal{M}) \), where \( \mathcal{M} \) is the size of the WCDS. Hence, for the distributed version of the NNDC, from the motion function, the movement of each neuron \( du \) is only related to the states of its neighbours. Therefore, the message overhead is the same as that of the AREA. The computing time of energy has the time complexity \( O(n^2) \), and the iterations is limited in a very small value. Therefore, the time complexity of the sequential neural network model is \( O(n^2) \). The time complexity of CENTRAL is \( O(n^2) \) too. However, the time complexity of distributed version of the neural network model is \( O(d) \), where \( d \) is the degree of a node, as the calculation only concerns with the neighbours of the current node, according to the motion function in Formula (3).

VII. CONCLUSIONS

A new Hopfield neural network for minimising the weakly connected dominating set for the self-configuration of wireless sensor networks was presented in this paper. We explored the convergence procedure by simulation of the sequential model, and tested the model on complete graphs to observe the robustness of parameters in the model. We also tested normal mesh graphs, and got approximately optimal results. Further, we presented an improved hill climbing algorithm with directed convergence, heuristic adjustment, and threshold compensation. The experimental results show the later (NNDC) achieves much better performance in the quality of solutions and efficiency. Moreover, the NNDC is robust in parameters \( A \) and \( B \). The model achieves better results than CENTRAL when the transmission radius is small, and the same performance as CENTRAL, when the transmission radius increases. The time complexity of the distributed NNDC is \( O(d) \), where \( d \) is the degree of a node, while the time complexity of CENTRAL is \( O(n^2) \). We investigated the effect of transmission radius on the size of WCDS. The results indicate that it is important to select a suitable transmission radius to make the network have a good stability and a long lifespan. The proposed model can be used on sink nodes in sensor networks, so that a sink node can inform the nodes to be a clusterhead in the WCDS obtained by the algorithm.

The message overhead is \( O(\mathcal{M}) \), where \( \mathcal{M} \) is the size of the WCDS. For the distributed version of NNDC, the message overhead is the same as that for the algorithm AREA.

Acknowledgement We really appreciate editors and reviewers for their valued comments.

REFERENCES


