WSN-ANN: Parallel and Distributed Neurocomputing with Wireless Sensor Networks

Gursel Serpen, Jiakai Li, Linqian Liu, and Zhenning Gao

Abstract—This paper proposes wireless sensor networks as a parallel and distributed computing platform for neurocomputing. The proposal entails leveraging the existing wireless sensor networks technology to serve as a hardware-software platform to implement and realize artificial neural network algorithms in a fully parallel and distributed computation mode. The study describes the proposed parallel and distributed neurocomputing architecture, which is named as WSN-ANN, and its use as a hardware platform on a case study. A Hopfield neural network, which is configured to solve the minimum weakly connected dominating set problem, is embedded into a wireless sensor network. Simulation study results indicate that the proposed computing platform based on wireless sensor networks, WSN-ANN, is feasible and promising to serve as a parallel and distributed neurocomputer.

Keywords - artificial neural network; wireless sensor network; parallel and distributed processing; parallel hardware; parallel and distributed computer architecture; Hopfield neural network; graph optimization

I. INTRODUCTION

A truly parallel and distributed hardware implementation of artificial neural network algorithms has been a leading and ongoing quest of researchers. Numerous attempts to realize the hardware implementation in silicon proved to be too challenging particularly when the scale of the implementation increased to dimensions at par with the real-life problems. The very large scale integration (VLSI) technology has been unable to deliver a truly parallel and distributed realization of artificial neural network algorithms at large scales. One of a few larger scale VLSI technology based hardware realization of neural networks is discussed in [27] from purely an academic perspective. Authors report a neural network design with 6144 spiking neurons and 1.57 million synapses. In another recent study reported in [7], a neural network with up to 10K neurons has reportedly been realized and plans for implementing a neural network with 100K neurons and 100 million synapses was discussed as part of the future plans. There are also other studies that report essentially very small-scale VLSI hardware realizations of artificial neural networks [15,23,29].

The realization of neural algorithms through VLSI on silicon hardware as integrated circuits (IC) has been a mainly academic exploration and failed to score any notable real-life or commercial success to date.

Simulations on parallel architectures (in traditional computer architecture sense) fail to scale with the size of the neural network since both time and space complexities quickly reach a level that is beyond what is affordable. Even if multiple processors of a parallel computing platform update a multiplicity of neurons in a given neural network and specialized concurrency techniques perform or facilitate certain operations in parallel (i.e. matrix algebra) the spatio-temporal cost of pure simulation is still insurmountable as extensive empirical evidence indicated. Perhaps lack of simulations for truly large-scale neural networks in either academic literature or in use anywhere (with one recent exception [1], which apparently has been subject to controversy as to validity of its findings [17],) is a testimony to the fact that the option of pure simulation is severely constrained for any practical utility.

There have been numerous attempts to build specialized computing platforms based on a mix of hardware and software components. The resultant computing systems were byproducts of different techniques drawn from software or hardware domains to essentially speed up or to accelerate computations. One such paradigm entails software models running on high-end supercomputer grade computing platforms like the Blue Brain [16] or Beowulf cluster [11]. Both projects require very expensive supercomputing hardware and highly specialized software.

Field programmable gate array (FPGA)-based approach forms the basis of a yet another implementation paradigm where primary software routines are implemented in hardware for significantly accelerated computing. Although FPGA-based approach offers great flexibility, practitioners often struggle to establish the correct system balance between processing and memory while also dealing with a harder programming aspect compared to software.

Another implementation paradigm is the custom-built hardware, which has been tried many times without notable success owing mainly to the fundamental problems which are inherent to application specific integrated circuits (ASIC). It has proven to be a major challenge, as evidenced by the lack of an operational system deployed in the field, to deal with the issue of deciding how much of the neural network functionality should be realized through hardware, which typically lead to the optimization of performance but the loss of flexibility.

A parallel and distributed implementation of artificial neural networks algorithms in a form somewhat similar to how biological neural networks exist in a brain could facilitate true real-time computation of solutions for very large-scale and real-life problems. This paper strives to address this outstanding challenge and consequently
proposes a new computer architecture based on wireless sensor networks for neurocomputing. The proposed neurocomputer is conceived to serve as a parallel and distributed processing platform for artificial neural networks to maximally enable their computational potential for real-time and large-scale context.

II. WIRELESS SENSOR NETWORKS FOR PARALLEL AND DISTRIBUTED NEUROCOMPUTING

A. Wireless Sensor Networks as a PDP System for ANNs

We propose a wireless sensor network (WSN) as a parallel and distributed computer architecture for neurocomputing. A WSN may consist of tens or hundreds of thousands of (either homogeneous or heterogeneous) processing nodes. Computations associated with implementing an artificial neural network algorithm are composed of a very large number of similar and rather simple calculations, which can be performed in parallel and a distributed manner. Associating a processing node in a WSN with a neuron in an ANN will induce a maximally parallel and distributed computation structure.

The WSN for neurocomputing may be modified as follows. The processing nodes or motes in the WSN may be supplied grid or line power rather than portable or battery power. Furthermore, modified WSN motes don’t need sensors for neurocomputing unless the application requires sensory inputs. Accordingly, a modified wireless sensor network is composed of motes each of which has an on-board microcontroller with ROM and RAM, a radio, and an antenna. The WSN protocols for medium access, time synchronization, positioning and localization, topology management and routing can be employed as appropriate with minimal or no adaptation and modification in most cases [13,28].

Each mote in a modified WSN implements the computational model of at least one neuron through its microcontroller and stores the weights of its neuron in its local memory. It will receive the outputs of other neurons over the air through the wireless radio communications from other motes. It will send the output of its own neuron to other neurons housed in other motes through the wireless channel. A modified WSN mote can house more than one neuron if needed.

The proposed design envisions down to one ANN neuron per each WSN mote if maximum parallelism and distributed computing is desired. It can accommodate multiple neurons per WSN mote up to a number that is feasible through the processing power of the microcontroller which is onboard each mote as needed.

The modified WSN offers a highly flexible computing platform that can be configured on-the-fly and over-the-air as needed. The entire embedded neural network can be recast or redefined for its type, structure, topology, connections or parameters (weights) with minimal effort, cost and, perhaps more importantly, dynamically. This suggests that the modified WSN is a generic (rather than specialized or customized) hardware computing platform for artificial neural networks.

Figure 1. Synthesis of WSN and ANN to form WSN-ANN

It is a powerful combination when a modified WSN as described above with thousands or hundreds of thousands of motes is coupled with a distributed algorithm that implements a certain task that can be decomposed into a very large number of subtasks potentially with massive concurrency or parallelism for execution. Mapping such an algorithm to a modified WSN will result in truly parallel and distributed computation, and hence is well positioned to facilitate real-time solution of large-scale problems.

An artificial neural network can be embedded not only within a wireless sensor network (WSN), but also a wireless sensor and actuator network (WSAN) as in Figure 2. In fact, the resulting parallel and distributed processing hardware system through embedding an ANN in a WSAN then has sensing and actuating attributes, which are essential elements for adaptation and autonomous behavior or operation. Embedding of an ANN within a WSN leads to a computing system that is massively parallel and distributed and can “sense” its environment and potentially use the sensed data or information in its computations, which is a context- or environment-aware computing. Embedding an ANN within a wireless sensor and actuator network (WSAN) results in a computing system that can not only sense its environment, but affect it through either networked and discrete or on-board actuator(s), and that can lead to adaptive and intelligent closed-loop control among other prominent attributes.

B. Applicability to ANN Algorithms

Any artificial neural network algorithm can be embedded within a WSN. A non-exhaustive sample of ANN algorithms which can be embedded within a WSN includes feed-forward architectures like multi-layer perceptron (MLP) [31] and radial basis function (RBF), self-organization algorithms including Kohonen’s self-organizing map (SOM) [32], linear vector quantizer (LVQ) and similar, associative memory neural networks including Hopfield associative memory, bi-directional associative memory (BAM), adaptive resonance theory (ART) nets and its many derivatives, and recurrent neural networks including Elman, Jordan, Hopfield, and simultaneous recurrent neural nets (SRN), and time delay neural networks (TDNN). In fact, VLSI implementations of the entire suite of ANN algorithms are presented in Cichocki et al. [3], which can be readily leveraged to map any type of neural net algorithm to an
arbitrary topology of a wireless sensor net. In the next section, we will discuss and demonstrate, through a specific but highly complex example, the feasibility of how to embed an ANN algorithm (configured for a protocol level optimization task) within a WSN for a fully parallel and distributed realization.

![Diagram](image)

Figure 2. WSAN (WSN+Actuators+Sensors) and ANN

### C. Embedding a Hopfield Net into Modified WSN

This section presents an illustrative case for mapping a neural network to a modified wireless sensor network. The Hopfield neural network is chosen as an example case of artificial neural network algorithms since its setup, initialization, and computation cycles are among the most complex among neural algorithms. Hopfield network offers a distributed optimization algorithm for computation of a local optimum solution of a static optimization problem. The promise is a quick and local optimum solution, and scalability of the computation time with the increase in the size of the problem for a hardware-centric implementation that takes advantage of the high-degree of inherent parallelism [10]. A Hopfield neural network is a nonlinear dynamical system, whereby the definition of the continuous Hopfield neural network is as follows. Let \( z_i \) represent a neuron output and \( z_i \in [0, 1] \) with \( i = 1, 2, \ldots, K \), where \( K \) is the number of neurons in the Hopfield network. Then, \( E(z) = -\sum_{i=1}^{K} \sum_{j=1}^{K} w_{ij} z_i z_j + \sum_{i=1}^{K} f^{-1}(z_i) - \sum_{i=1}^{K} b_j z_i \) (1)

is a Liapunov function for the system of equations defined by

\[
\frac{du_i(t)}{dt} = -u_i(t) + \sum_{j=1}^{K} w_{ij} z_j(t) + b_i \quad \text{and} \quad z_j = f(u_j),
\] (2)

where \( w_{ij} \) is the weight between neurons \( z_i \) and \( z_j \) subject to \( w_{ij} = w_{ji} \) and \( w_{ii} = 0 \), \( b_i \) is the external bias input for node \( z_j \), and \( f(z) \) is a nonlinearity - typically the sigmoid function with positive slope steepness value represented by \( \lambda \). Note that the second term in the Liapunov function vanishes for very large positive values of the parameter \( \lambda \) for cases where the activation function is sigmoidal shaped.

One form of equations that are suitable for parallel and distributed hardware realization for the Hopfield neural network is given as [5]:

\[
\frac{dv_i(t)}{dt} = -\mu_i \left( v_i(t) - \tanh \left( \sum_{j=1}^{K} w_{ij} v_j(t) + \theta_j \right) \right) / T
\]

for \( i = 1, 2, \ldots, K \) and \( \mu_i > 0 \), (3)

where \( v_i \) is the output value for neuron \( i \), \( w_{ij} \) is the weight from the output of neuron \( j \) to input of neuron \( i \), \( \theta_j \) is the external bias value for neuron \( i \), \( \mu_i \) is a parameter with a strictly positive real value, \( T \) is a time varying (computational) temperature parameter, and \( n \) is the number of neurons in the network. This set of equations can be discretized, say using Euler’s rule. Consequently, we obtain the following discrete-time model of the Hopfield neural network computations:

\[
v_i^{(k+1)} = v_i^{(k)} - \mu_i \left( v_i^{(k)} - \tanh \left( \sum_{j=1}^{K} w_{ij} v_j^{(k)} + \theta_j \right) / T \right)
\]

for \( i = 1, 2, \ldots, K \) and \( k = 1, 2, 3, \ldots \) (4)

The ANN given in Equation 4 is ready to be mapped to the WSN topology.

The procedure of embedding a Hopfield neural network within a modified wireless sensor network can be stated as follows. Consider a wireless sensor network with \( N \) motes and a Hopfield neural network with \( K \) neurons. Assuming \( K < N \) and maximally parallel and distributed computations are desirable; each mote is assigned a single neuron. Each mote computationally implements a single neuron, i.e. calculates the \( k \)-th iteration value for the discrete-time equivalent of the dynamics given in Equation 4. The same mote also allocates the storage needed for the weight vector for the neuron, bias and threshold, nonlinearity slope, and others. The weight vector, bias and threshold terms for a given neuron residing on a given mote are initialized to the values obtained through resolving problem specific error function with the Liapunov function in Equation 1. In general, any given neuron can talk to any other neuron in the network (through multi-hop communications over the WSN). This establishes the required connectivity of the Hopfield neural network as dictated by the specific optimization problem energy function. Connections to neurons on one-hop neighbor motes will be direct or without any intermediaries. Connections to neurons residing on motes that are not one-hop neighbors of the host mote will be over multiple hops.

### III. SIMULATION STUDY

This section presents a simulation study to demonstrated the feasibility of the proposed design and profile its computational and communication complexities, and performance. The proposed computer architecture, WSAN-ANN, is simulated for feasibility of serving as a parallel and distributed processing platform for neurocomputing. The simulation study profiles the performance of a wireless sensor network embedded with a Hopfield neural network.
configured for the minimum weakly connected dominating set (WCDS) problem.

A. Minimum Weakly Connected Dominating Set Problem

A dominating set (DS) is a subset of nodes or vertices in a graph (a typical abstract model of a sensor network) such that each node is either in DS or has a neighbor in DS [2]. A connected dominating set (CDS) is a connected DS, that is, there is a path between any two nodes in CDS which does not use nodes that are not in CDS. It might be favorable to have few nodes in the CDS. This is known as finding the minimum connected dominating set (MCDS) in an arbitrary graph. More formally, the minimum CDS problem is defined as follows: given an (arbitrary) undirected graph find a CDS with a minimum number of nodes. The minimal weakly connected dominating set problem is of interest since it is comparably easier to compute. A subset of nodes $S$ in a graph $G=(V,E)$ is weakly connected if the weakly induced subset $Sw$ is connected; $Sw$ consists of $S$ and all the neighbors of $S$ while the edges in $Sw$ are from $E$ where each has at least on endpoint in $S$. There are number of challenges to face for WSN implementations given that computing the minimal WCDS is NP-hard. Accordingly, finding a WCDS that is “close” to the minimum (i.e. an approximation rather than an exact solution) might be desirable in most cases. Furthermore, the solution must be local since global solutions are impractical for dynamic networks. A minimum WCDS based on the graph model of a wireless sensor network is the primary underlying framework for the sensor network infrastructure.

B. Mapping Minimum WCDS Problem to Hopfield Network

The minimum WCDS problem [2] can be mapped to the Hopfield network dynamics as follows. Assume a graph is a set of $N$ vertices, $V$, $i=1,2,...,N$, and up to $K=N^2$ edges, $e_{ij}, i,j=1,2,...,N$, where some of the edges may not exist. Consider a neural network $N$ neurons where outputs of neurons are represented by $z_1,...,z_N$. Each neuron in the neural network will be mapped or correspond to a vertex in the graph. An active neuron ($z_i=1$) will represent that the vertex to which it is mapped is selected for inclusion in the dominating set. All other neurons whose corresponding vertices in the graph have an edge to the vertex mapped to this active neuron should be inactive ($z_i=0$). Also for any neuron that is inactive, exactly one neuron should be active among all neurons which represent its adjacent vertices. These statements can be captured by the following energy function under the assumption that neuron output values converge to limiting values in the interval $[0,1]$:

$$E = \frac{1}{2} g_a \sum_{i=1}^{N} \sum_{j=1}^{N} e_{ij} z_i z_j + \frac{1}{2} g_b \sum_{i=1}^{N} \left( 1 - \sum_{j=1}^{N} e_{ij} z_j \right) (1 - z_i)$$

\[ (5) \]

where it is required that $g_a, g_b \in R^+$. The energy term has a globally minimum value of zero when both constraints are satisfied or, equivalently stated, when both terms assume a value of zero. The first term has a minimum value of zero when all adjacent neurons of an active neuron are inactive. The second term is zero when exactly one neuron is active among all the adjacent neurons for a given inactive neuron. This error function can be associated with the generic Liapunov function in Equation 1 for Hopfield dynamics to derive values for the weights, biases and threshold for each of the neurons in the network, which is then considered to have been configured to solve the minimum WCDS problem.

C. Simulation Environment and Setup

The simulation study will employ the TOSSIM software tool [26], an emulator for TinyOS-Mica based sensor networks. TOSSIM emulates the TinyOS networking stack at bit-level or packet-level granularity, which is user selectable.

The distributed application code implementing Hopfield neural network configured for the minimum WCDS problem was written using the nesC (network embedded systems C) programming language to execute on the TinyOS operating system for wireless sensor networks [14]. TinyOS version 1.1.x was used to implement this project on Windows XP OS with Cygwin user interface. TinyOS also requires the Java SDK: the default version of Java SDK (1.4.2) was utilized.

Default wireless sensor network protocols as incorporated into the TinyOS 1.1.x stack with the exception of the routing protocol were used. TinyOS employs Berkeley-MAC (B-MAC) [22] medium access protocol which is based on Carrier Sense Medium Access (CSMA). For time synchronization, TinyOS employs the Flooding Time Synchronization Protocol (FTSP), which is an ad-hoc, multi-hop time synchronization protocol [18]. The default routing protocol in TinyOS is an ad-hoc multi-hop routing protocol, which is based on a shortest-path-first algorithm with a single destination node and active two-way link estimation. Instead of this default flooding-based multi-hop routing protocol, we use a modified version, where each mote stores its neighborhood information in the local memory to serve as the routing table. The message is forwarded via this routing table.

WSN motes are assumed to be randomly and uniformly distributed across a rectangular two-dimensional plane. Each mote is assigned a unique ID. Numbering for mote IDs starts at 0 and increases by 1 for each additional mote in the network. There are two kinds of motes in the sensor network being used to implement the neural network, the supervisory (aka reference or gateway) mote and generic motes. Generic motes perform single neuron computations while the only supervisory mote (across the entire wireless sensor network) acts as a “supervisor” – it monitors, collects, processes and broadcasts the global information related to neural network computations. The supervisory mote is responsible broadcasting start and end times of neural computation phases including initialization, convergence or neural network dynamics update, solution detection and analysis, and re-initialization as needed.

Neural network computations take place while the mote is awake and is subjected to the schedule of the mote on
which the neuron resides. Depending on the number of neighbors and the time interval, the sleep time for motes varies from 95% to 98%. We use a timer on every node to manage timing during the network updating. The timers, which are synchronized network wide, are set to fire independently or asynchronously for all aspects of the WSN operation including neighborhood discovery, initialization, and periodic activity including neural network computations in normal operation. Each mote wakes up from sleep once every one time period (which is set to 0.1, 0.2, 0.5 and 1s for the simulation study) through its dedicated timer. Timers are synchronized network-wide through the FTSP. Accordingly, each neuron output gets updated once per 1 time period asynchronously of other neuron outputs.

The simulations were performed for WSNs with mote counts of 10, 50, and 100, and repeated 10 times for each mote count. WSN motes are randomly and uniformly distributed over a two-dimensional rectangular area with dimensions of 100×100 units. Accordingly, simulations ran for radio transmission radius values of “empirical”, 25, and 50 units where feasible. To maintain the sensor network connectivity, radio transmission radius values of “empirical” and 50 units are applicable for networks with 10 and 50 motes; while radio transmission radius values of “empirical”, 25, and 50 are feasible for 100-mote WSNs.

The values for the two constraint weight parameters of Hopfield neural network error function in Equation 5 in conjunction with the bound in [24,25] which guarantee that local and global minima correspond to solutions of the minimum WCDS problem are derived as $g = 2.0$ and $g_b = 1.0$. The reader is referred to the reference 24 for details of how these values can be derived, which is not presented herein due to space limitations.

Performance metrics evaluated included the quality of solutions, the number of packets exchanged during the neural computation (message complexity), and the time complexity (as measured in the number of timer ticks or wake-up periods). The quality of solutions is the size of WCDS, in another words, it is the count of motes (active neurons in the corresponding graph) that form the WCDS. The message complexity is calculated by counting all the packets received by individual motes in the network.

Calculating time complexity of the computations associated with the neural network implementation requires a somewhat elaborate procedure since TOSSIM does not model the execution time for any code segment. In other words, from TOSSIM’s perspective, a piece of code runs instantaneously. The computations associated with implementing neuron dynamics on each mote are not costly due to simplicity of calculations as well as the parallel and distributed nature of these computations. Accordingly the basic idea of counting the number of timer periods needed to converge to a solution by the Hopfield neural network can be employed a reasonable measure to assess how the time complexity is affected by the increase in the WSN mote count and the length of the time interval associated with the sleep-wakeup periods.

### D. Simulation Results

Simulation was performed using TOSSIM-packet (which realizes hardware emulation at the packet-level granularity) and results are presented in Table I. The foremost observation relates to the feasibility of the proposed neurocomputing architecture. Simulation results demonstrate that the proposed WSN-ANN neurocomputer is able to compute reasonable quality solutions of the minimum WCDS problem. Furthermore, the WCDS solution quality (as measured by the size of the graph node count included in the solution) is at par with those of solutions computed by other prominent distributed algorithms reported in the literature [2].

<table>
<thead>
<tr>
<th>Mote Count</th>
<th>Radio Range</th>
<th>Time Interval</th>
<th>WCDS Size (No of nodes)</th>
<th>Computation Time (timer ticks)</th>
<th>Message Complexity (# of packets)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Mean Deviation</td>
<td>Mean Deviation</td>
</tr>
<tr>
<td>10</td>
<td>Empirical</td>
<td>0.1</td>
<td>3.6</td>
<td>12.1</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.2</td>
<td></td>
<td>11.3</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.5</td>
<td></td>
<td>10.1</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Empirical</td>
<td>0.1</td>
<td>4.2</td>
<td>12.0</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.2</td>
<td></td>
<td>11.5</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.5</td>
<td></td>
<td>10.4</td>
<td>1.4</td>
</tr>
<tr>
<td>50</td>
<td>Empirical</td>
<td>0.1</td>
<td>8.1</td>
<td>27.1</td>
<td>0.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.2</td>
<td></td>
<td>27.1</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.5</td>
<td></td>
<td>27.1</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>Empirical</td>
<td>0.1</td>
<td>10.7</td>
<td>64.1</td>
<td>2.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.2</td>
<td></td>
<td>64.1</td>
<td>2.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.5</td>
<td></td>
<td>64.1</td>
<td>2.2</td>
</tr>
<tr>
<td>100</td>
<td>Empirical</td>
<td>0.1</td>
<td>15.8</td>
<td>60.1</td>
<td>4.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.2</td>
<td></td>
<td>60.1</td>
<td>4.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.5</td>
<td></td>
<td>60.1</td>
<td>4.5</td>
</tr>
<tr>
<td></td>
<td>Empirical</td>
<td>0.1</td>
<td>9.5</td>
<td>62.1</td>
<td>4.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.2</td>
<td></td>
<td>62.1</td>
<td>4.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.5</td>
<td></td>
<td>62.1</td>
<td>4.2</td>
</tr>
</tbody>
</table>

He et al. [8] used a Hopfield neural network to compute a locally optimal solution for the minimum WCDS problem on a PC for a graph model of a wireless sensor network. In other words, the solution was computed by a Windows application: they used a non-distributed implementation of Hopfield neural network executing on a PC to calculate the solutions for WCDS: more importantly, no aspect of wireless sensor networks such as wireless protocols appears to have been modeled. They report that their results for WCDS size are comparable to those of the best performing algorithm CENTRAL [2,8] in the relevant literature. A comparison of our results with those reported in [8] also shows that the solution quality is comparable. This is a very encouraging finding since it suggests that the proposed PDP neurocomputer is not only feasible but also promising.

To facilitate performance comparison with the distributed WSN realization, centralized Hopfield neural network was also simulated for neuron counts of 10, 50, and 100 for the minimum WCDS problem. For each mote count, the simulation was repeated 10 times and the data was modeled using the t distribution. The Hopfield neural network settings, such as the two constraint weight parameters in...
The leading element of complexity is the messaging increase, as indicated in Table I. This is expected since the mote count is made for mote counts of 100 and 10, which is a 10-fold increase. The messaging complexity is approximately 30-fold when the comparison is made for the fully-distributed and local realization of WSN-ANN.

In Table II, the difference in WCDS size is less than 10% for the smaller radio range of 25, which reduces the number of one-hop neighbors and therefore results in less wireless communication medium congestion. The difference increases to 51% for a mote count of 100 and radio range of 50. This indicates that by controlling the radio range it may be possible to reduce the difference in WCDS sizes.

In general, for distributed and local computation on a wireless sensor network, the data and information exchange among the neurons will not be deterministic as it is for the centralized implementation: some data packets will be lost or delayed due to lack of time synchronization, medium access collisions, multi-hop routing, and sleep schedules (the latter is if adopted). This will result in neuron computations with outdated, expired or repeat data in some cases. Accordingly, the net effect appears to emerge as deterioration in the quality of solution.

As the “Computation Time” measurements in Table I suggest, the time complexity is not affected by the “Time Interval” or the “Radio Range” by an appreciable amount. The mote count however is the determining factor in this respect as the mean Computation Time increases from 12 timer ticks for a 10-mote WSN to a range of 20 to 40 timer ticks (or sleep-wakeup periods) for the 100-mote WSNs. The increase in the message complexity is more pronounced. This increase is approximately 30-fold when the comparison is made for mote counts of 100 and 10, which is a 10-fold increase, as indicated in Table I. This is expected since the leading element of complexity is the messaging requirements. The empirical findings in Table I also indicate that the messaging cost appears to increase faster than the time cost. We follow up these empirical findings based on the simulation with a complexity-based analysis and projections in the next section to demonstrate the scalability of the proposed PDP neurocomputer based on the WSN herein.

E. Scalability of Proposed Architecture

The time, space and messaging complexities will determine the scalability of the proposed WSN-ANN neurocomputing framework for the minimum WCDS problem. Particularly the messaging complexity plays a critical role in scalability since the time complexity is also dependent on it. The space complexity is linear in the size of the WSN (e.g. each mote maintains two vectors or arrays as data structures, namely the weight vector for the embedded neuron and the neural network output vector where the size of both grows linearly in the count of motes.

Assuming a mote count of 100,000 for the WSN, the Hopfield neural network would have 100,000 neurons. The entire 100,000 neurons of Hopfield net are distributed to 100,000 motes of the WSN on a one-neuron-per-mote basis. Noting that each neuron is located on a dedicated mote, the amount of processing for neuron dynamics and memory space to be allocated for the neuron weight vector (including the bias and threshold parameters) and the output value of the neuron by the on-board microcontroller are negligibly small for all practical purposes. Assuming that the Hopfield neural net is configured to compute in asynchronous mode, each mote can update the on-board neuron dynamics and calculate the output in an asynchronous manner. Accordingly all 100,000 neurons can be updated in parallel and fully distributed fashion. Each neuron update will require receiving neuron output values from neighboring motes (per the minimum WCDS problem definition), which introduces the communication or messaging cost. Consequently, the dominant factor for constraining scalability here is the messaging complexity since on the order of 10^{15} messages will need to be exchanged.

There is substantial parallelism in communications since many messages can be communicated concurrently due to limited trans-receiver range for a given mote and possible employment of multiple channels for communications. If, on the average, motes are clustered into groups with 10 motes per group, then there will be 10,000 such groups for a WSN with 100,000 motes. For instance, in a typical minimum WCDS context for topology control, each dominating node will be communicating with a specific group or set of non-dominating nodes, which will form a cluster. What this means is that 10,000 inter-group or inter-cluster communications can happen simultaneously without jamming or causing interference to each other’s message exchanges. This would mean that now 10^{15} messages need to be exchanged in a sequential mode. Although specific numbers are highly technology, protocol and topology dependent, assume that, for the sake of illustration, each message requires an average communication or transmission time of 10^{-3} second (which is 1 micro second). Then the messaging cost will be on the order of 10^{12} seconds. If the WSN employs multi-channel communications and there are

<table>
<thead>
<tr>
<th>Mote Count</th>
<th>Radio Range</th>
<th>PC Simulation (Non-distributed)</th>
<th>WSN-ANN Simulation (Distributed)</th>
<th>T Statistic</th>
<th>P Value</th>
<th>Percentage Difference in WCDS Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Empirical</td>
<td>3.8</td>
<td>0.9</td>
<td>3.6</td>
<td>1.3</td>
<td>0.4</td>
</tr>
<tr>
<td>50</td>
<td>3.9</td>
<td>0.6</td>
<td>4.2</td>
<td>1.9</td>
<td>-0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>50</td>
<td>Empirical</td>
<td>6.3</td>
<td>0.8</td>
<td>8.1</td>
<td>1.1</td>
<td>-4.2</td>
</tr>
<tr>
<td>100</td>
<td>4.5</td>
<td>0.7</td>
<td>8.4</td>
<td>1.1</td>
<td>-4.6</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>8.8</td>
<td>1.0</td>
<td>10.7</td>
<td>2.7</td>
<td>-2.1</td>
<td>1.0</td>
</tr>
<tr>
<td>14.6</td>
<td>1.1</td>
<td>15.6</td>
<td>2.7</td>
<td>-1.3</td>
<td>0.9</td>
<td>8.2</td>
</tr>
<tr>
<td>50</td>
<td>6.3</td>
<td>0.5</td>
<td>9.5</td>
<td>0.8</td>
<td>-10.7</td>
<td>1.0</td>
</tr>
</tbody>
</table>
10 such channels for messaging, then this number is further reduced to 10^6 seconds or 16.7 minutes. Although this is still a relatively considerable cost for one convergence episode of the Hopfield net dynamics, it can be managed through further fine tuning and optimizations. For instance, transmitting neuron outputs only when they change and not each time they are recalculated can cut down drastically the number of related messages. Another possibility for potentially significant reduction in messaging complexity is the scenario where more than one neuron can be embedded in a given mote. The messages among those neurons co-located in the same mote would not need to be wirelessly communicated to each other. A single message could also be employed to wirelessly communicate the outputs of all neurons co-located in the same mote to other neurons on different motes. Similarly, fine tuning the topology, the medium access control protocol, and the routing protocol will also help reduce the communication latency further.

IV. CONCLUSIONS

This paper presented a parallel and distributed computer architecture for neuro-computing based on wireless sensor networks, which is named as WSN-ANN. The proposed computer architecture, due to its hardware-level fine-grain parallelism, is poised to leverage the full potential for parallel computation inherent to the artificial neural network algorithms. Feasibility and computational promise of the proposed neuro-computer WSN-ANN was demonstrated through a simulation study. A Hopfield neural network configured as a static optimizer for a graph-theoretic optimization problem was embedded within a wireless sensor network. The WSN-ANN was able to compute reasonable quality solutions for the weakly-connected dominating set problem. A complexity-based scalability analysis suggested that the proposed neuro-computer, WSN-ANN, has the promise to scale up and offers strong prospects for real time computation of solutions for large-scale problems.

REFERENCES
