2007

Embedding Distributed Learning Algorithms in Wireless Ad-Hoc Control Networks

A. Desmet
University of Wollongong, ad011@uow.edu.au

F. Naghdy
University of Wollongong, fazel@uow.edu.au

M. Ros
University of Wollongong, montse@uow.edu.au

Publication Details
Embedding Distributed Learning Algorithms in Wireless Ad-Hoc Control Networks

Abstract
With the advances in soft computing techniques and agent technologies, the concept of home ambient intelligence is becoming more and more realistic. Living in a building that adapts itself to the users and assists them in reducing their energy consumption is now within reach. The main technical barrier comes from hardware: servers and industrial control networks do not fit in a house. With the availability of dedicated wireless solutions and low-cost small computation units, the platform to implement task distribution in a control network is now feasible and cost efficient. This paper explores the possibilities of fitting a distributed learning algorithm for home ambient intelligence in a wireless network of sensors and actuators, driven by very limited microcontrollers. The chosen hardware platform is the WACNet: Wireless Ad-hoc Control Network. The concept of WACNet is introduced and the test-bed developed for its study is explained. The fuzzy learning algorithm is then introduced and its implementation is studied. The results of a test are provided and some conclusions are drawn, mainly focusing on accuracy and the algorithm’s response to different rule selection criterions.

Disciplines
Engineering | Physical Sciences and Mathematics

Publication Details
Embedding Distributed Learning Algorithms in Wireless Ad-Hoc Control Networks
Antoine Desmet∗, Fazel Naghdy† and Montserrat Ros‡
School of Electrical, Computer and Telecommunications Engineering
University of Wollongong, Wollongong NSW 2522 Australia
∗Tel: +61-2-4221 3575, E-mail: ad011@uow.edu.au
†Tel: +61-2-4221 3398, Fax: +61-2-4221 3236, E-mail: fazel@uow.edu.au
‡Tel: +61-2-4221 3133, E-mail: montse@uow.edu.au

Abstract—With the advances in soft computing techniques and agent technologies, the concept of home ambient intelligence is becoming more and more realistic. Living in a building that adapts itself to the users and assists them in reducing their energy consumption is now within reach. The main technical barrier comes from hardware: servers and industrial control networks do not fit in a house. With the availability of dedicated wireless solutions and low-cost small computation units, the platform to implement task distribution in a control network is now feasible and cost efficient. This paper explores the possibilities of fitting a distributed learning algorithm for home ambient intelligence in a wireless network of sensors and actuators, driven by very limited microcontrollers. The chosen hardware platform is the WACNet: Wireless Ad-hoc Control Network.

The concept of WACNet is introduced and the test-bed developed for its study is explained. The fuzzy learning algorithm is then introduced and its implementation is studied. The results of a test are provided and some conclusions are drawn, mainly focusing on accuracy and the algorithm’s response to different rule selection criteria.

I. INTRODUCTION
Home ambient intelligence is a terminology that groups every application intended to assist the users of a building, in a seamless and non-intrusive way. Building automation systems, ancestors of ambient intelligence, have slowly evolved by implementing more and more complex but still hard-coded (and therefore rigid) behaviours. Since 1980, some ambient intelligence research projects actually focused on intelligent and learning systems [1,2,3]. These were the first systems where the user did not need to configure the system. Most of these projects rely on multi-agent systems using fuzzy-logic learning algorithms. Still, in most of the proposed solutions, requirements such as: the deployment of an expensive Fieldbus network, or a central server permanently online, makes it unrealistic for a home and low-budget own-installation.

Advances in wireless networks and the widespread availability of low-cost small computation units has provided the hardware platform to realize highly distributed computation of data. Such hardware platforms would increase the overall flexibility, scalability and modularity. Besides, this hardware is ideal for an easy self-installation of a home ambient intelligence system. The major gap between theory and realization lies in managing to run soft computing algorithms (that usually run on powerful computers) on a platform made of several communicative microcontrollers.

The concept of Wireless Ad-hoc Control Network (WACNets) has been developed with the goal of providing a framework for highly distributed, intelligent wireless control networks [5,6]. The nodes in WACNets have some computation capacity and memory which can be used to run virtually any distributed application. The overall network structure is ad-hoc, easing the deployment and increasing the scalability. A WACNet is an ideal solution for home installation because it is inexpensive, requires no configuration or setup, can run on batteries and is wireless and infrastructure-less.

This study aims at exploring the possibility of implementing a distributed home ambient intelligence system in a WACNet structure for real-world applications. The focus is on bridging the gap between the complex, resource-demanding learning algorithms and the constraints of a network of restricted nodes, requiring task distribution. The main objective is, after a learning period, to locally predict the output of a sensor, given the state of the system.

The remainder of the paper is structured as follows.

Fig. 1. WACNet nodes and network structure
The concept of WACNet is first presented. The home ambient intelligence application is introduced and the challenges linked to the implementation in a WACNet are listed in the third section. The next part describes the fuzzy-learning algorithm, its physical implementation and all its parameters. A test bed is presented in the fifth section, followed by the results of different experiments. Finally, conclusions are drawn on various aspects, focusing on the results of the learning algorithm and the issues related to coping with the insufficient amount of memory.

II. THE HARDWARE PLATFORM: WACNETS

The concept of Wireless Ad-hoc Control Networks (WACNets) [5,6] represents a new stage in the evolution of distributed control and monitoring. It explores a framework for organic, evolutionary and scalable integration of a large number of nodes with sensing and/or actuation, local intelligence and control, data processing and communication capabilities.

A WACNet consists of a set of geographically distributed intelligent and heterogeneous nodes. In this study, each node consists of a processing unit (ATMEL Mega 32), and a ZigBee wireless communication unit (MaxStream XBee). The processing unit performs various tasks, along with communication control. This makes the WACNet architecture ideal for task distribution, as it is composed of several small units with high communication capacity.

The network’s topology is a mesh of many clusters. A cluster comprises of a cluster head, which serves as a routing gateway for the slave nodes that surrounds it. As the ZigBee Routers implements an ad-hoc routing protocol, it provides the WACNet with transparent multi-hop message transmission over the whole network. It is also self-organizing and self-healing, infrastructure-less, as every cluster head (acting as ZigBee Router) extends the reach of the network.

III. AMBIENT INTELLIGENCE & WACNETS

The final goal of this research project is to create an ambient intelligence system with a multi-agent core. It will assist the inhabitants of a building in reducing the amount of resources (such as electricity and water) they consume. The multi-agent core comprises two types of agents: learning agents, which learns and forecasts the needs in resources, and policy agents, which apply hard-coded scheduling and reduction rules, based on the information supplied by the learning agents. The focus of this paper is on the learning agents, and especially the distribution of a learning algorithm. The ultimate goal of the learning agents is to extract a set of fuzzy rules that predicts the state of a given sensory value, provided the state of some of the remaining sensors in the network. This is done by polling at regular intervals the whole network, to learn which are the laws ruling the system and which variables are involved.

As mentioned above, this is not a very difficult problem to solve, given a powerful workstation, adequate software and a FieldBus control network. In this paper, the challenge is to distribute this gigantic task in several relatively limited networked microcontrollers.

The most challenging technical aspects are discussed in the following list:

A. Communications

The ZigBee RF standard provides the network with self-organizing ad-hoc routing. A message can be delivered anywhere in the network with a latency of a few ms. Cluster Heads collect the sensory data of the whole cluster in a single frame, to reduce the overall amount of messages transferred. This technique avoids bottlenecks and congestion.

B. Distribution

A WACNet is totally decentralized. All nodes are hierarchically equal, and must process all their data locally. Therefore, the learning agent embedded in every node will only focus on rules that apply to the sensors and actuators connected to its hosting node.

C. Memory space and computation capacity

The speed of the processor (16 MHz) is not the major problem. The program memory (32kb) is also well sufficient to hold all the drivers and a small fuzzy-learning algorithm. The main issue is sharing the 2kb of RAM between the program variables, the serial transmit and receive buffers, and the fuzzy rules table. It is also impossible to store the learning data sets permanently, without extending the memory.

IV. THE LEARNING ALGORITHM AND ITS IMPLEMENTATION

The learning agents will use a fuzzy logic-based learning algorithm. This has proven to best cope with
the changing and irregular behavior of human beings. This learning algorithm first considers rules such as: IF Var1 is \( n_1 \) AND Var3 is \( n_3 \), THEN Var5 is \( n_5 \).

The following equation gives \( S \), all possible rules,

\[
S = \prod_{u=0}^{p} n_{var} \left( x_n \right)
\]

(1)

Where \( n_{var} \) is the number of sets that represent the possible values of a given sensor \( x_n \).

As new data sets are made available, this algorithm increments all the matching rules’ weight field by the current weight. In this study, the weight corresponds to the changing and irregular behavior of human beings. This phenomenon is illustrated in table 1.

The possible rules and their associated weights would be the combinations of all fuzzy sets:

- \( S_1:1\), \( S_2:1 \rightarrow 0\% \) ( = min(40,0))
- \( S_1:2\), \( S_2:1 \rightarrow 0\% \) ( = min(90,0))
- \( S_1:1\), \( S_2:2 \rightarrow 90\% \) ( = min(40,90))
- \( S_1:2\), \( S_2:2 \rightarrow 90\% \) ( = min(90,90))

The algorithm will not consider the two first combinations, as they are totally irrelevant (weight = 0). For the two last combinations, the occurrence counter will be incremented and the value 90 will be added to the previous sum of weights. The division of the two previous fields will provide an average firing strength for the given rule.

This simple principle is quite efficient and accurate, but requires refinement from a rule optimization algorithm. A typical undesirable behavior is the generation of several similar rules, to cope with all the possible states of irrelevant variables (as proven in previous simulations [4]). Practically, considering a given law and an unrelated light switch, the system will have to promote a rule that fires with the light switch on and another one when it’s off: because all terms in the rule are ANDed, they all must be found true for the rule to apply. Thus, the simulation results were correct but not optimum, as the amount of selected laws, globally, was over five times what was required. This phenomenon is illustrated in table 1.

Here, the main issue comes from the platform’s limitations: the agent must optimize the rules. It is also impossible to keep every single law that fired: there is only room for the most essential ones. Consequently, the agent must also be capable of performing rule selection, requiring some kind of rule relevance metrics. The experiment presented next focused on the selection process.

As seen above, this fuzzy learning technique must be completed by a rule selection tuning method, to meet memory constraints. Though it has been proven that these tasks can be achieved by genetic algorithms,

Considering a network of two sensors \( S_1 \) and \( S_2 \) with two fuzzy sets each (1 and 2), a possible sensory data could be \( S_1:1\), \( S_2:2\) [40% 2: 90%], \( S_2:1\) [0% 2: 90%].

<table>
<thead>
<tr>
<th>rank</th>
<th>Weight</th>
<th>Rule:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100%</td>
<td>if daylight IS BRIGHT AND pres. is FALSE AND w-mach IS OFF AND water cons. IS HIGH THEN light IS DARK</td>
</tr>
<tr>
<td>2</td>
<td>93%</td>
<td>if daylight IS DARK AND pres. is FALSE AND w-mach IS OFF AND water cons. IS LOW THEN light IS DARK</td>
</tr>
<tr>
<td>3</td>
<td>89%</td>
<td>if daylight IS MED AND pres. is FALSE AND w-mach IS OFF AND water cons. IS NORM THEN light IS DARK</td>
</tr>
<tr>
<td>4</td>
<td>85%</td>
<td>if daylight IS DARK AND pres. is FALSE AND w-mach IS OFF AND water cons. IS HIGH THEN light IS DARK</td>
</tr>
<tr>
<td>5</td>
<td>81%</td>
<td>if daylight IS BRIGHT AND pres. is FALSE AND w-mach IS OFF AND water cons. IS LOW THEN light IS DARK</td>
</tr>
<tr>
<td>6</td>
<td>76%</td>
<td>if daylight IS DARK AND pres. is FALSE AND w-mach IS OFF AND water cons. IS NORM THEN light IS DARK</td>
</tr>
<tr>
<td>7</td>
<td>75%</td>
<td>if daylight IS MED AND pres. is FALSE AND w-mach IS OFF AND water cons. IS HIGH THEN light IS DARK</td>
</tr>
<tr>
<td>8</td>
<td>74%</td>
<td>if daylight IS MED AND pres. is FALSE AND w-mach IS OFF AND water cons. IS HIGH THEN light IS DARK</td>
</tr>
<tr>
<td>9</td>
<td>73%</td>
<td>if daylight IS MED AND pres. is FALSE AND w-mach IS OFF AND water cons. IS NORM THEN light IS DARK</td>
</tr>
<tr>
<td>10</td>
<td>69%</td>
<td>if daylight IS BRIGHT AND pres. is FALSE AND w-mach IS OFF AND water cons. IS LOW THEN light IS DARK</td>
</tr>
<tr>
<td>11</td>
<td>69%</td>
<td>if daylight IS DARK AND pres. is FALSE AND w-mach IS OFF AND water cons. IS NORM THEN light IS DARK</td>
</tr>
<tr>
<td>12</td>
<td>69%</td>
<td>if daylight IS MED AND pres. is FALSE AND w-mach IS OFF AND water cons. IS LOW THEN light IS DARK</td>
</tr>
<tr>
<td>13</td>
<td>68%</td>
<td>if daylight IS BRIGHT AND pres. is FALSE AND w-mach IS OFF AND water cons. IS NORM THEN light IS DARK</td>
</tr>
<tr>
<td>14</td>
<td>67%</td>
<td>if daylight IS MED AND pres. is FALSE AND w-mach IS OFF AND water cons. IS NORM THEN light IS DARK</td>
</tr>
<tr>
<td>15</td>
<td>65%</td>
<td>if daylight IS MED AND pres. is FALSE AND w-mach IS OFF AND water cons. IS NORM THEN light IS DARK</td>
</tr>
<tr>
<td>16</td>
<td>64%</td>
<td>if daylight IS DARK AND pres. is FALSE AND w-mach IS OFF AND water cons. IS NORM THEN light IS DARK</td>
</tr>
<tr>
<td>17</td>
<td>63%</td>
<td>if daylight IS DARK AND pres. is FALSE AND w-mach IS OFF AND water cons. IS NORM THEN light IS DARK</td>
</tr>
<tr>
<td>18</td>
<td>62%</td>
<td>if daylight IS BRIGHT AND pres. is FALSE AND w-mach IS OFF AND water cons. IS NORM THEN light IS DARK</td>
</tr>
</tbody>
</table>

In this experiment, the light (consequent) was set to 100% DARK when no presence was detected, regardless of the amount of light in the room, or the state of the washing machine or the water consumption. These 18 rules are totally redundant and are equivalent to a single rule: when presence is FALSE then light is DARK.
they require a lot of computation and memory, usually look up previous data sets, and can be slow to converge. This is why this solution cannot be considered here.

There are two metrics easily available for rule selection: average weight and number of firing occurrences.

V. EXPERIMENT DESCRIPTION

A microcontroller is programmed to emulate four sensors: two I/O and two analog. All possible combinations of the system in this case make 36 fuzzy rules. The environment emulator produces a semi-random set of states with different probabilities, so that the reaction to rare events can also be considered. The states change every 2 seconds. Each port of the emulator is connected to a WACNet node’s sensor port. The node acquires and locally fuzzifies the data, and transmits the fuzzified data on request. A cluster head collects and condenses the data from the whole cluster into a single frame, and sends it back to the requesting node hosting the learning agent, every two seconds.

A control test is done until 3,000 global rule match occurrences are reached, and every firing rule and its metrics are saved. This test reveals the correct weight of each rule, and the average frequency of occurrences.

The reading of the control test reports 20 valid rules. These rules are represented by the 20 first rows of the Table 2, and can be classified into the three following categories:

-1: A unique state perfectly matching a single rule, which fires very frequently (probability = 0.15),
-2: A set of events that are categorized in 2 categories, each potentially matching 9 rules. These rules are fired less often (0.06<p<0.04) with a lower weight (from 50 to 15%). This constitutes the most usual state (global probability = 0.75)
-3: A rare state that happens less that 2% of the time, but which also perfectly matches a single rule.

This control test also happened to mark rules that do not apply to any state programmed in the event emulator. This happens when the node reply delay is too long and some variables happen to change during the data acquisition period. The collected data instead reports an erratic period. The collected data instead reports an erratic

<table>
<thead>
<tr>
<th>weight</th>
<th>Occur.</th>
<th>Prob.</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5%</td>
<td>0</td>
<td>125</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1%</td>
<td>0</td>
<td>128</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5%</td>
<td>0</td>
<td>132</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1%</td>
<td>0</td>
<td>138</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5%</td>
<td>0</td>
<td>146</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1%</td>
<td>0</td>
<td>150</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

This table illustrates the data set. The darkest row is the case 1 event, the gray correspond to the various case 2 states, and the light grey row represents the case 3 rule. The remaining rows characterize various error rules.

It has been decided that the learning agent would only hold space for 20 rules. To get more stable results, a two-level rule table is introduced. Among the twenty slots, all new and recent rules will stay in one of the 12 slots available in probation zone. Not until they manage to reach a threshold value of firing occurrences (15, in this case), will they access the reliable zone (8 slots). Only the rules from the reliable zone will be applied. Every new rule that reaches the threshold value systematically overwrites the weakest rule in the probation area, to leave the system open to new rules.

As mentioned in the description of the algorithm, the weakest rule in a zone can be determined either from its firing strength or its number of firing occurrences. The two eviction criteria have been tested in separate experiments. The experiments stopped when over 1,000 rule matches occurred.
VI. EXPERIMENT RESULTS AND ANALYSIS

Both of the experiments returned correct values. The system was able to self-configure a selection process and determine which rules were relevant, according to the chosen metrics. Not a single error rule appeared in the reliable zone. This is due to the two-level structure of the rule table. The errors do not happen often enough to even secure a stable slot in the probation area. Therefore, they never get up to the threshold and access the reliable zone.

- The test using the firing occurrences as a metric for rule eviction only selected the rules that fired strictly more than 4% of the time, regardless of their weight: the lowest weight was 18%, which denotes a moderately relevant rule. It mostly consisted of the rules from the categories 1 and 2.
- The test using rule weight as the eviction criterion kept only the rules with a weight greater than 25%. In this test, the selected rules had various firing probabilities.

Using either the rule weight or rule firing occurrence as a metric for rule eviction gives the system a specific orientation:

- Rule firing occurrences selection produces better results for the most commonly found states of the system. On the other hand, it is incapable of “remembering” rare events.
- Rule weight selection builds a rule table which is better capable of dealing with all situations found in the learning data sets. On the average, it will give slightly more imprecise results, but the accuracy will depend less on the event’s probability. In other words, this rule table will have a “rough idea” of every case.

VII. CONCLUSIONS

This paper has introduced the goal of this project, and described a particular part of it: the implementation of a learning agent on a distributed wireless control system. It has developed the principle and implementation of a fuzzy-learning algorithm, and presented the hardware platform, the WACNet.

The experiments have proven that a restricted distributed learning algorithm can run on a network of communicative microcontroller-based nodes and obtain acceptable results.

The experiments have also shown that it is possible to select only the most accurate rules in order to save memory, whilst keeping accuracy. The two-level rule table has efficiently filtered very exceptional and erratic learning data sets, and added stability to the selected rule set.

For the application considered (i.e. human behavior learning), the rule weight selection will be preferred. As the agents do not require a precise forecast of resources consumption, it is preferable to opt for a system handling every single behavior of the users, even the less common ones.

REFERENCES


Fig. 4. Management of the two-level rule table