Adaptive Systems for Smart Buildings Utilizing Wireless Sensor Networks and Artificial Intelligence

By

Blerim Qela

A Thesis Submitted to the

Faculty of Graduate and Postdoctoral Studies

In Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

In

Electrical and Computer Engineering

School of Electrical Engineering and Computer Science

University of Ottawa

Ottawa, Ontario, Canada

DEDICATION

To the Universal Light of Knowledge

And

All the People of Good Will

ABSTRACT

In this thesis, research efforts are dedicated towards the development of practical adaptable techniques to be used in Smart Homes and Buildings, with the aim to improve energy management and conservation, while enhancing the learning capabilities of Programmable Communicating Thermostats (PCT) – "transforming" them into smart adaptable devices, i.e., "Smart Thermostats". An Adaptable Hybrid Intelligent System utilizing Wireless Sensor Network (WSN) and Artificial Intelligence (AI) techniques is presented, based on which, a novel Adaptive Learning System (ALS) model utilizing WSN, a rule-based system and Adaptive Resonance Theory (ART) concepts is proposed. The main goal of the ALS is to adapt to the occupant's pattern and/or schedule changes by providing comfort, while not ignoring the energy conservation aspect. The proposed ALS analytical model is a technique which enables PCTs to learn and adapt to user input pattern changes and/or other parameters of interest.

A new algorithm for finding the global maximum in a predefined interval within a two dimensional space is proposed. The proposed algorithm is a synergy of reward/punish concepts from the reinforcement learning (RL) and agent-based technique, for use in small-scale embedded systems with limited memory and/or processing power, such as the wireless sensor/actuator nodes. An application is implemented to observe the algorithm at work and to demonstrate its main features. It was observed that the "RL and Agent-based Search", versus the "RL only" technique, yielded better performance results with respect to the number of iterations and function evaluations needed to find the global maximum. Furthermore, a "House Simulator" is developed as a tool to simulate house heating/cooling systems and to assist in the practical implementation of the ALS model under different scenarios. The main building blocks of the simulator are the "House Simulator", the "Smart Thermostat", and a placeholder for the "Adaptive Learning Models". As a result, a novel adaptive learning algorithm, "Observe, Learn and Adapt" (OLA) is proposed and demonstrated, reflecting the main features of the ALS model. Its evaluation is achieved with the aid of the "House Simulator". OLA, with the use of sensors and the application of the ALS model learning technique, captures the essence of an actual PCT reflecting a smart and adaptable device. The experimental performance results indicate adaptability and potential energy savings of the single in comparison to the zone controlled scenarios with the OLA capabilities being enabled.

ACKNOWLEDGEMENTS

I fall short of words to express gratitude and appreciation to my supervisor Dr. Hussein T. Mouftah, for his valuable advice, encouragement, moral support and outmost professionalism shown during my research and preparation of this thesis. I thank you for your patience; devotion and guidance in helping me reach here today. It was an honor to have a chance to work under your supervision.

I would like to thank Dr. T. Yeap and Dr. R. Yu for their valuable comments and inputs during my research. I would also like to thank Dr. J. Yao and Dr. J. Zhao for their moral support and understanding. In addition, I would also like to thank my colleague Mark Trigg for proof reading my thesis.

I am deeply grateful to my parents for raising me with love and care, and to my wife Edi for her love, moral support, patience and understanding during my studies. And to my children Jeta, Rrezarta and Endrit…I cannot turn back time; however, I will try to make up for the times that I have missed spending with you.

Above all, I give thanks to God for giving me strength and inspiration to follow my dream.

Table of Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
ALS	Adaptive Learning System
AMI	Advanced Metering Infrastructure
AmI	Ambient Intelligence
ANN	Artificial Neural Networks
ART	Adaptive Resonance Theory
ASCENT	Adaptive Self-Configuring Sensor Networks
ASHRAE	American Society of Heating, Refrigerating and Air-Conditioning Engineers
CODA	Congestion Detection and Avoidance in Sensor Networks
CPU	Central Processing Unit
DEVS	Discrete Event System Specification
DR	Demand Response
EEPROM	Electronically Erasable Programmable ROM
EGU	Electricity Generating Utility
FEL	Future Event List
GA	Genetic Algorithms
GUI	Graphical User Interface
HVAC	Heating, Ventilation and Air Conditioning
KB	Knowledge-Base
LEACH	Low Energy Adaptive Clustering Hierarchy
LED	Light Emitting Diode
LCD	Liquid Crystal Display

MAC	Medium Access Control
OOD	Object Oriented Design
OLA	Observe, Learn and Adapt
OSI	Open System Interconnection
PCT	Programmable Communicating Thermostat
PLL	Phased Locked Loop
PSO	Particle Swarm Optimization
RAM	Random Access Memory
RISC	Reduced Instruction Set Computing
ROM	Read Only Memory
RL	Reinforcement Learning
RTC	Real Time Clock
SCADA	Supervisory Control and Data Acquisition
SQL	Structured Query Language
TAG	Tiny Aggregation
TCP	Transmission Control Protocol
TOU	Time of Use
UART	Universal Asynchronous Receiver Transmitter
UDP	User Datagram Protocol
VAV	Variable Air Volume
WSN	Wireless Sensor Networks

List of Symbols

A_c	Adapting vector
L	Learning vector
A^*	Set of adapting vectors
L^*	Set of Learning vector
S_g	Set of wireless sensor groups
w	Weight associated with learning element
ϵ	Tolerance (based on which particular weight elements are evaluated)
β	Weight multiplier coefficient
λ	Learning rate
Θ	Resulting value of adapted element
A_f	Airflow rate
H_t	Heater
β_h	Heater stages used
T_n	Room temperatures
ΔT	Temperature difference
ρ_n	Airflow offsets
μ_n	Heater offset
K_b	Knowledge base function used to optimize zone control
c	Specific heat capacity of the air at constant pressure
d	Density of the air
k	Thermal resistivity of materials
q_h	Heated air supply
R_{eq}	Equivalent thermal resistance of the house
R_{wall}	Thermal resistance of a wall
R_{window}	Thermal resistance of a window

l	Wall Thickness
L_{wall}	Thickness of walls
L_{window}	Thickness of windows
K_{wall}	Thermal coefficient of walls
K_{window}	Thermal coefficient of windows
A_{wall}	Area of walls
A_{window}	Area of windows
λ_{th}	Thermal conductivity
T_{room}	Room (indoor) temperature
$T_{external}$	External (outdoor) temperature
M_f	Air mass flow rate through heater
M_{air}	Mass of air
V	Volume of the house
T_{heat}	Heat temperature set point
T_{cool}	Cool temperature set point
T_{room}	Room temperature set point
Q_{heat}	Heat gain of a house
Q_{cool}	Cool gain of a house
Q_{losses}	Heat losses of a house
Q_{cool_losses}	Cool losses of a house
$T_{room\_increment}$	Room temperature increment
$T_{room\_decrement}$	Room temperature decrement
ΔT_h	Difference of heat and room temperature
ΔT_l	Difference of room and external temperature
ΔT_c	Difference of room and cool temperature
ΔT_m	Difference of external and room temperature

CHAPTER 1

1.1 Introduction

The need for energy efficient and intelligent systemic solutions has led many researchers around the world to investigate and evaluate the existing technologies in order to create solutions that would be adopted in near future intelligent homes and buildings [HAG08] [CHA08]. The emergence of powerful embedded micro-computer systems, and wireless sensor networks (WSN), provides a good ground for in-depth research and adaptation of existing intelligent technologies and concepts, while exploring the new ones.

The above initiatives have motivated and led to many scientific endeavors and contributions to our society, where the existing state-of-the-art intelligent technologies and concepts are used in integration of many intelligent systems towards a new era of "Smart Homes and Buildings" [RED06][RIC06]. The leading edge technology in the area of intelligent systems and smart sensor networks are an essential part of our everyday life. Their evolution will help us to better utilize our energy resources (i.e. energy saving initiatives), and enhance our way of living. Indeed, many governments and utilities are interested to better utilize electricity, and encourage initiatives leading towards the development of intelligent systems. Thus, numerous research groups are closely involved in bringing forward efficient "Smart Home and Building" systems for our living environments, furthering the research in intelligent and automatic control systems.

Programmable thermostats [DOU94][DOU09] are used widely for automatic control of temperature and humidity, and nowadays extend into a Programmable Communication Thermostat (PTC) [MEI08]. PCTs are equipped with LCD user interfaces, push button controls, and wireless interface, for communications and network capability to a multitude of sensors/actuators, offering a variety of options for controlling thermal comfort and/or other appliances [RED07][KUS07]. Today's PCTs are able to communicate with home appliances, electricity generating utility (EGU) meters (i.e., Smart Meters) helping in the peak load control (demand response initiatives) and offering efficient use of energy resources. The utilization and integration of PCT devices and multiple WSN into Home

Automation Systems contribute to the accomplishment of "Smart Homes and Buildings" [YUP07].

Many sophisticated fuzzy control systems, WSN schemes [KAN10] and devices for "Smart Homes and Buildings" are being investigated and evaluated by researchers. The concepts of intelligent systems are being investigated and tested in many heating ventilation and air conditioning (HVAC) applications, such as for the utilization of sensors and intelligent thermostat systems for multi-zone HVAC control systems.

Furthermore, the concepts of "Smart Thermostats" are being investigated in order to come up with systemic solutions which are adaptable, energy aware, and easy to use. As described in [MEI08], "intelligent thermostats" are used in many smart homes and buildings. However, the learning "intelligent thermostats" mentioned in [MEI08] are programmable thermostats, which are capable of learning the occupant's pattern by interaction (i.e. when the user changes the set point temperature, the thermostat remembers it), and it uses those parameters in the next daily schedule. However, this does not address the problem of learning the schedules and/or pattern changes without user interactions. Thus, compared to an actual "Smart Thermostat" only represents a programmable thermostat capable of remembering (i.e., store in memory) the user preferred set point temperatures or daily schedules (similar to any typical PCT). Nevertheless, it requires the user interaction/input to be modified, and it also does not address the problem of intelligent zone controlled environment. The effect of human behavior is also described in [MEI08], where about 25 to 50 percent of the common U.S. households utilize the programmable thermostats as an on/off switch. Furthermore, homes relying on the programmable thermostats for energy savings, consumed more energy than the homes where the occupants set the temperatures manually. Moreover, in many cases, constant programming of the thermostat poses a "hassle", for many users. The results from the survey about "the amounts of time users operate the air conditioner/heater" [MEI08] indicate only an occasional usage pattern. Additionally, it shows that even the "tech-savvy users" were dissatisfied with the complexity of the programmable thermostats and its user interface.

In many present commercial buildings, the amount of sensors used for the purpose of energy efficiency is relatively small, and in residential buildings is typically reduced furthermore, and has little use of embedded intelligence [ARE05] i.e., single programmable

thermostats. The energy use in U.S. alone, for heating/cooling of the commercial and residential buildings is around 38 percent [ARE05]. From this total energy usage, the energy consumption for the commercial buildings is 28 percent, for residential buildings is 43 percent, and the rest of consumption is for the water heating, lighting, and miscellaneous usage. The energy usage for space heating and cooling in residential buildings is the highest (43 percent). Thus, utilizing multiple wireless sensors/actuators, applying Artificial Intelligence (AI) techniques and Ambient Intelligence (AmI) perspective for energy aware smart environments, is very significant.

Moreover, the advantages of using multi-sensor versus single sensor systems for control of HVAC are also described in [LIN02]. Although the energy performance and comfort is improved in a multi-sensor system, the described system utilizes only one actuator to control the HVAC. The use of a multi-actuator system for control of air dampers in individual rooms, in addition to the control of the HVAC, would prove to be more efficient solution. Since the preferences of occupants and their perceived comfort might be different, the optimal thermal comfort of a multi-zone environment i.e. use of multiple actuators to control the air dampers into individual zones based on the occupants' preferences, is advantageous.

Furthermore, as the demand for electricity continues to grow, transition to the smart grid/smart metering environment, among others, as described in [VOJ08] requires "smart devices and in-home energy management systems, such as PCTs capable of making intelligent decisions based on smart prices". The peak load curtailment, demand response (DR) and Time-Of-Use (TOU) rates are among many factors considered in smart grid initiatives, where the importance of investing in dynamic and flexible designs and smart devices is essential [VOJ08]. The importance of peak shifting (i.e., redistribution of a task away from the time of peak demand) and energy conservation via a computer-based system to measure and manage energy consumption is described in [WIL06]. The peak shifting is essentially a pre-programmed thermostat schedule based on EGU (i.e., utility) TOU rates and DR energy management incentives. However, the cost of the described monitoring and control system is estimated to be $2,500 per home, requires user input/interaction and does not reflect adaptive learning capabilities.

With the advancement and broad application of AI models in technology, the emergence of adaptive learning systems [WIR00] is obvious. The rule-based expert systems are widely

used in technology and are considered to be the best option for building knowledge based systems [SHE08], where the formulation of knowledge is based on the expert's opinion, represented by simple production rules. Typically with an IF-THEN structure, where multiple conditions can be joined by 'AND' and 'OR' keywords, facilitating representation of relations, recommendations, and strategies for different scenarios. In a nutshell, structure of the rules-based expert systems consists of the knowledge base, database of facts, inference engine (i.e. links facts and knowledge), explanations facilities (i.e. enables user to ask and see how a particular conclusion is reached), user interface and the user. One of the disadvantages of the rule-based expert systems is their incapacity to learn and their slow response (if a very large set of rules is considered).

Other AI models, such as, the Adaptive Resonance Theory (ART) [CAR87a], Fuzzy ART [CAR91], self-organizing Neural Networks, Genetic Algorithms (GA), Fuzzy Logic and Particle Swarm Optimization (PSO) are also being used in cases of machine learning endeavors, such as the unsupervised learning system and optimization problems. In particular the ART1 and Fuzzy ART models are being used in different technological and biological applications involving unsupervised learning for binary and analog input patterns. Moreover, the emergence of 'intelligent agents' [POS08], which perceive their environment through sensors and act upon that stimulus via actuators, is obvious as well.

In many real-world applications, the use of only one AI model (in most cases) would not suffice to bring forward the best systemic solutions. While the combination of different techniques has led to the emergence of more sophisticated intelligent systems, known as hybrid intelligent systems (i.e., which combine at least two intelligent technologies).

1.2 Motivation and Objective

The need for efficient environmental controls is becoming apparent as the EGUs are looking to better utilize and manage power, its use and control of the peak load demands, while the consumers are looking for comfort and conservation. To better manage the ever-increasing energy demands, electricity costs, and environmental impacts, many governments and companies are looking for more efficient solutions to the existing problem. The purpose of Advanced Metering Infrastructure (AMI) and DR initiatives is to assist EGUs to meet their

energy needs, by introducing TOU prices/rates, with the intention of encouraging users to shift part of their electricity use to off peak hours. Thus, allowing the customers to conserve and save (on high electricity costs), and respectively, assist the EGUs to better manage the peak load demands.

On the other hand, the role of a programmable thermostat is to provide the consumer with a means to manage and reduce energy use, while accommodating their every day schedules. The changing schedules, comfort set point temperatures, preferences, needs and patterns of consumers, are different. The constant interaction and/or programming of the thermostat might not prove flexible, nor optimal enough with respect to comfort and the conservation aspect of it, due to the available features, difficulty of use or programmability, limited number of sensors/actuators nodes, lack of system interaction and/or communication, lack of systems intelligence, etc.

The PhD Thesis objective is to investigate, and address the above issues via simulation, experiment and development of an "*Adaptive Learning System"*, providing a smart and adaptable energy management systemic solution for intelligent buildings. Thus, bringing forward a "Smart Thermostat" for optimal energy management in intelligent buildings – a hybrid intelligent system solution, which utilizes WSN and AI techniques to learn and adapt. The "Smart Thermostat" uses sensors to observe/monitor, actuators to control, and AI techniques to learn and adapt. It does not require constant programming input of the occupant(s), learns and adapts to the occupant(s) preferences, schedules and/or pattern changes, offers zone-controlled environment solution, and responds to utility DR and TOU price rates incentives. Thus, further improving the comfort and conservation of energy management in intelligent buildings, and simultaneously offering new insights and solutions that can help towards the advancement of science, in general.

1.3 Research Description

The envisioned concept of *"Adaptive Systems Utilizing WSN and AI"* encompasses a WSN consisting of numerous "intelligent agents" (smart sensor/actuator nodes) and a central controller unit. It offers intelligent energy management for smart homes and buildings by using the rule-based expert system and adaptive learning principles. Thus, it is an "energy aware" system solution, capable of acting, learning and/or adapting to the occupant(s)

preferences, schedules and pattern changes, rates of heating/cooling of different rooms (i.e., zone control). When applied to the environmental control problem, AI techniques enable the overall system to adapt and learn from the system dynamics, utilizing a rule-based expert system strategy and adaptive learning principles.

The rule-based expert systems cannot offer the best possible solution alone. They have a drawback, which is their dependence on prior knowledge and also the amount of time required to match the rule with the memory. Using only one AI technique, in this case, would not render a flexible solution. Thus, the goal is to investigate and utilize other AI concepts, such as AI models based on the unsupervised learning strategies.

In the proposed research, AI techniques, such as the rule-based expert systems and unsupervised learning with biological motivations will be investigated, in order to provide the overall system flexibility to learn and adapt to new knowledge, without destroying the existing knowledge. Thus, offering a hybrid intelligent system solution for conservation of energy and comfort zone adjustment. Because our main objective is to create an adaptive learning system, existing models, such as ART, self-organizing neural networks, expert systems and potential new techniques are of vital importance and shall be investigated to determine the best possible choice for the hybrid intelligent system being proposed and considered. One critical element of this research is to combine several AI techniques leading to an "*Adaptive Learning System"* – a hybrid intelligent system capable of learning in our living environments.

The Hybrid Intelligent System (refer to Figure 1.1) proposed, is based on the rule-based expert system and unsupervised learning techniques (where the problem is how to adapt to new knowledge without destroying the existing knowledge). The "Smart Thermostat" core controller unit is equipped with distributed sensors (i.e., intelligent agents), which use the rule-based expert system and ART concepts to learn and/or adapt. The role of sensors in the proposed scheme is to monitor the environment (i.e. temperature, occupancy detection, air flow, etc.), while the actuators are used to control the heater/cooler stages and for the adjustment of the air flow (i.e. via air dampers) in a zone-controlled environment. The proposed system enables the comfort zone adjustment, i.e., the control of heating/cooling of individual rooms and/or of the entire house; and is capable of processing inputs to and/or from the EGU (i.e. Utility/Smart Meter) to the core controller unit.

The research approach taken emphasizes a scalable solution, considering only a few inputs / outputs and simple user interface (UI) at the initial stages of research, and afterwards emerging to a more complex system with multiple sensors, sources of information, and a variety of output types.

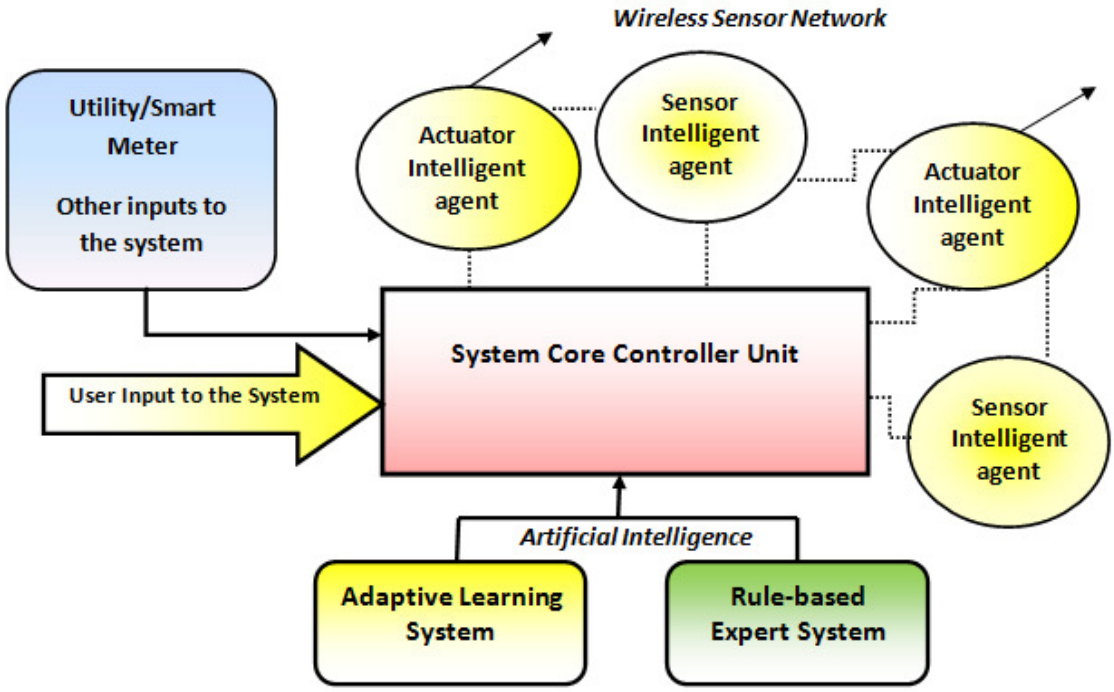


Figure 1.1 - Hybrid intelligent system

The aim is to create an adaptive system which enables intelligent power management in "Smart Homes and Buildings". Thus, the solution extends beyond a programmable thermostat, by proposing an adaptive hybrid intelligent system – a "Smart Thermostat", which does not require constant programming input by the occupant(s) and is capable of learning and adapting, while offering the optimal energy conservation and comfort of the occupant(s). In addition, "Smart Thermostat" communicates with the Smart Meter, providing efficient energy savings, and helping EGUs to better manage the peak load demand, by responding to TOU price rates, and DR incentives.

1.4 Summary of Contributions

In this thesis, the research efforts are devoted to the development of adaptive learning techniques to be used in Smart Homes and Buildings, with the aim to improve energy management (comfort and conservation aspect of it), and propose learning capabilities to current PCTs. The main contributions of this thesis are shown below:

- An Adaptive – Hybrid Intelligent System Solution utilizing Wireless Sensor Networks and Artificial Intelligence Techniques for Energy Management in Smart Homes and Buildings is proposed.
    - Analytical Model of the Adaptive Learning System (ALS) is proposed.
- A novel algorithm based on the Reinforcement Learning and Agent-based technique, for finding the global maximum in a predefined interval for use in small-scale embedded systems with limited memory and/or processing power, such as the wireless sensor/actuator nodes is proposed, implemented and demonstrated.
    - A specific application is implemented and developed to verify and confirm its performance.
- A House Simulator constructed with 'thoughtful consideration' for its use as an 'expert system shell', is proposed.
    - Analytical model of the "Simulation Engine" is presented.
    - House Simulator is designed and implemented in order to prove the envisioned concepts of:
        - "Adaptive Systems using WSN and AI" and
        - Its use for evaluation, implementation, and verification of new adaptive learning techniques for future "Smart Thermostats".
- A novel adaptive learning technique: "Observe, Learn and Adapt" for future "Smart Thermostats" using wireless sensors and AI is implemented and demonstrated based on the proposed Adaptive Learning System model.
    - Knowledge-base technique is implemented and applied in conjunction with the OLA algorithm, and demonstrated as an essential part of it.

1.4.1 List of Publications

1. B. Qela and H.T. Mouftah, *"Intelligent Systems for Energy Management in Wireless Sensor-based Smart Environments,"* in Sustainable Green Computing: Practices, Methodologies and Technologies, ed. W. Hu and N. Kaabouch, IGI Global, USA, 2012.
2. B. Qela and H.T. Mouftah, *"An Adaptable System for Energy Management in Intelligent Buildings,"* in Proc. of 2011 IEEE International Conf. on Computational Intelligence for Measurement Systems and Applications (CIMSA 2011), pp.40-46, Ottawa, Ont., Canada, Sep. 19-21, 2011.
3. B. Qela and H.T. Mouftah, *"Observe, Learn and Adapt (OLA) - A new algorithm for Smart Homes, using Wireless Sensors and Artificial Intelligence"*(Under review).
4. B. Qela and H.T. Mouftah*, "Synergy of Reinforcement Learning and Agent-based Techniques, for Finding Optimal Solution in a Predefined Interval,"* in Proc. of the Summer Computer Simulation Conference (SCSC'10), Ottawa, Ont., Canada, July 12, 2010.
5. B. Qela and H.T. Mouftah, *"Simulation of a House Heating System using C# - An Energy Conservation Perspective,"* in Proc. of 23rd IEEE Canadian Conference on Electrical and Computer Engineering (CCECE'10), pp. 1-5, Calgary, AB, Canada, May 2010.
6. B. Qela and H.T. Mouftah, *"Synergy of Wireless Sensor Networks, Ambient Intelligence and Artificial Intelligence for Energy Management in Smart Homes and Buildings,"* 2nd Annual Wisense Workshop, Queen's University, Kingston, ON, Canada, May 13, 2010.
7. B. Qela, G. Wainer and H.T. Mouftah, *"Simulation of Large Wireless Sensor Networks Utilizing Cell-DEVS,"* in Proc. of 2009 Winter Simulation Conference (SCSC'09), pp. 3189-3200, Austin, TX., US, Dec. 2009.
8. B. Qela and H.T. Mouftah, *"Wireless Sensor Networks for Energy Management in Intelligent Buildings,"* 1st Wisense Workshop, University of Ottawa, Ottawa, ON, Canada, July 27, 2009.

1.5 Thesis Outline

The organization of subsequent sections of thesis is as follows: A survey of background material is provided in Chapter 2. In Chapter 3 a methodological approach, analytical modeling, and the key steps and synopsis of the proposed solution are described. In Chapter 4 a new algorithm for finding the global maximum of a function in a predefined interval for small-scale embedded systems with limited memory and processing power is presented, as a result of applying the concepts of the Reinforcement Learning (RL) and agent-based techniques. A specific application was developed and implemented in order to verify and establish the benefits of applying the above mentioned technique. Chapter 5 describes and demonstrates the simulator model, design architecture, and its aim to be used as an 'expert system shell' for further stages of research. Initial results and discussion of the simulator are presented in this section. In Chapter 6 a novel algorithm 'Observe, Learn and Adapt' (OLA) for smart homes, utilizing wireless sensors and artificial Intelligence concepts is presented. The implementation fundamentals, validation and performance results showing OLA's feasibility for adaptable learning thermostats is shown via simulation results and performance evaluation; necessary improvements to the existing simulator ('expert system shell') are depicted and presented as well. In Chapter 7 the concluding remarks and findings of the research efforts are presented. Recommendations for future research are also discussed. Appendices include additional valuable information related to the thesis. Appendix A introduces a simulation model of WSN by implementing the Topology Control Algorithm using the Cell-DEVS (Discrete Event System Specification) formalism. Further, in Appendix B, focus is to design and emulate a few potential features of a *Heater-Cooler System Prototype* by experiment (utilizing hardware and firmware tools), which could be beneficial for the interested reader. In Appendix C, the complete structure of Master and Daily clusters used for OLA algorithm are included. Whilst, in Appendix D confidence interval calculations used for statistical analysis of OLA results are included for completeness.

CHAPTER 2

INTELLIGENT SYSTEMS AND WIRELESS SENSOR NETWORKS

The development of intelligent electronic sensing devices, powerful embedded microcontrollers and wireless communication devices is a foundation for new advanced sensor devices, which can monitor actuate, compute and communicate, yet are small in size and cost [ELK08][HAR07][HON10]. The wireless sensor devices have the capability to self-organize into the so-called WSN. Their ability to sense diverse variables of interest, such as the temperature, humidity, pressure, airflow, occupancy, sunlight and other, could greatly improve the limitations of the existing and future energy management systems; many governments and EGUs are interested in finding solution to manage the ever-increasing energy demand, electricity costs and environmental impacts.

The Energy Management in Intelligent Buildings, by utilizing WSN, AmI, Advanced Control Systems and AI are key elements that embody the concept of "*Intelligent Buildings"* by striving to make them more adaptable, autonomous and aware of our environment; yet flexible and intelligent to sense, actuate, compute and evolve into *"Adaptive Systemic Solutions"* - adaptable, re-configurable systems, which act and adapt by exploiting wireless sensor/actuator network capabilities and system intelligence; enabling efficient energy management in our homes and buildings, enriching automation and control systems, and exploiting the renewable energy resources.

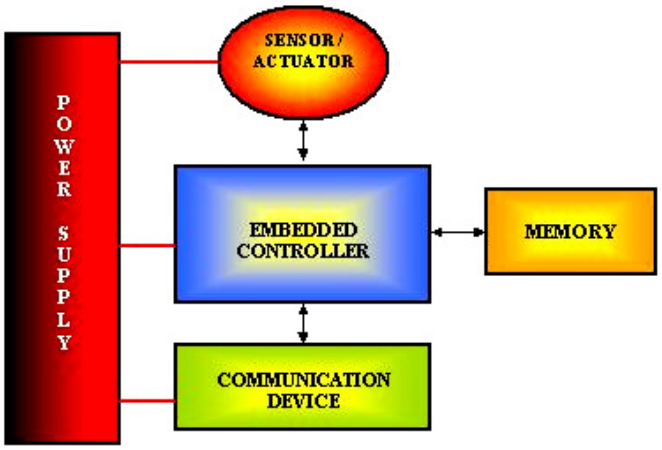
2.1 Wireless Sensor Networks

The emergence of powerful embedded micro-computer systems for WSN provides a good ground for creation of new smart sensor systems, which can be useful to further promote new scientific endeavors and enhance our lives. WSN are used in many different applications, such as medicine, transportation and urban monitoring, traffic control, military, environment and habitat monitoring, energy management, smart homes, automatic meter readings, industrial applications, telemedicine, etc [BYO08][CHE08][KAN10][NEW07] [SUH08][WU08]. The effectiveness of WSN is not just their monitoring, actuating,

computing and communications capabilities - with the added microcomputer processing power, analog and digital ports, transceivers and available memory, they have the capabilities to self-organize and communicate in the deployed area. Their processing power is limited, however WSN are usually deployed in large numbers and their load is shared accordingly.

2.1.1 Wireless Sensor Node Architecture

The main components of a Wireless Sensor Node as depicted in Figure 2.1 are the sensor/actuator, embedded controller (microcomputer) unit, power supply, memory and communication device (i.e., transceiver).



Sensor nodes are attractive for use in WSN due to their low cost, small size, versatility, easy deployment and energy efficiency. Sensor node architecture [HEA08][ZHE06], including its hardware and software platform chosen, depends on the application and requirements.

Due to the fact that sensor nodes have limited bandwidth, computing power and limited energy resources, one of the main constraints in WSN is the energy efficiency of sensor nodes (i.e., power consumption) [SAK07][ELI07][ZHE07]. The main sources of energy consumption of sensor nodes are the microcontroller and transceiver. Consequently, the energy consumption of sensor nodes is an essential factor when considering the wireless sensors for a specific application [YEA07]. In order to reduce the power consumption,

sensor nodes support different modes of operation, such as active, idle and sleep modes. The WSN Topology Control is one of the approaches used to solve the energy efficiency problem in sensor networks (refer to Appendix A for Simulation of Large WSN using Cell-DEVS).

2.1.1.1 Embedded Controller

Microcontrollers are most commonly as a core controller unit of the sensor nodes, due to their low cost, with memory build in, rich peripheral set and flexibility to connect to other devices, low power consumption in comparison to high-end processors (i.e., used in desktop computers) and rich set of peripherals including many input/output (I/O) ports; equipped with analog and digital converters, internal oscillators, comparators, timers, etc. In addition, microcontrollers can be programmed many times and there are plenty of debugging and development tools and kits (hardware and software) for prototyping and development purposes.

Some examples of microcontrollers used in sensor nodes are:

Texas Instrument

Texas Instruments MSP 430 is a 16-bit RISC microcontroller suitable for embedded application; powerful enough to handle the computational tasks that are required by wireless sensor nodes. MSP 430 is equipped with RAM (from 2 to 10 kB), A to D (analog to digital), D to A (digital to analog) converters, RTC (real time clock), many I/O ports and other interconnection possibilities.

Microchip PIC series

The PIC18F6720 series are *Microchip's* RISC based microcontrollers that contains 8-bit CPU core, RAM (4 kB), and most of the peripherals inside a single integrated circuit, including A to D, D to A, RTC, PLL, UART, many peripherals and I/O ports; enabling easy interface for sensor nodes.

Atmel Atmega128L

Atmega128L is an 8-bit micro (RISC architecture), feature rich, has 4 kB of RAM, 8 MHz oscillator, peripheral interconnection possibilities; intended for embedded applications - feasible for use in sensor nodes.

2.1.1.2 Memory

There are different types of memory used in microcontrollers:

- RAM (Random Access Memory)
  - Fast, mainly used for data
- ROM/EEPROM (Read only memory / Electronically Erasable Programmable ROM)
  - Used for external data storage and program code – erase and programming cycles are slow (in bytes)
- FLASH Memory
  - Program code memory
  - Enables many erase and programming cycles (in blocks/faster)

2.1.1.3 Transceivers

Transceiver is a communication device that consists of a transmitter and a receiver, which enables the two-way wireless communications among sensor nodes. The essential task performed by the transceiver in sensor nodes is be able to receive and transmit data via radio waves at a specific frequency (e.g. wireless sensor networks typically use frequencies between 433 MHz and 2.4 GHz). Main parts of the transceiver are the radio frequency building block (performs analog signal processing) and the baseband building block (performs the digital signal processing). There are many manufacturers that offer transceivers particularly suited for WSN, which incorporate all the necessary circuitry for transmit and receive operation.

Some examples of transceivers used in sensor nodes are:

RFM Monolithics RFM TR1000

- Designed for short range wireless communications
- Frequency ranges 868 MHz and 916 MHz
- Supports short-range radio communications (up to 115.2kb/s)
- Radiated power 1.5 dBm

Chipcon CC1000

- Designed for very low power and low voltage wireless sensor applications
- Wide frequency ranges (300 – 1 GHz)
- Programmable output power

Ember EM2420

- Wide Frequency ranges 868 MHz, 915 MHz and 2.4 GHz
- Data rate of 20, 40 and 250kb/s respectively
- Radiated power -0.5dBm
- Tx mode (22.7mA), Rx mode (25.2mA)

2.1.1.4 Sensor Node Power Supply

Batteries are the main source of power for the sensor nodes. For the purpose of tiny sensor nodes, batteries should be small in size, preferably rechargeable (via means of vibrations, solar and other), and with high capacity to operate for long periods of time. The capacity of the battery needs to withstand different modes of operations (active mode or high power, idle or sleep modes with low power consumption). Due to the fact that throughout the operation of the sensor node, the voltage level of batteries drops, delivering marginal power which might impair the sensor nodes' readings, the need for DC-to-DC converters is needed in order to overcome this problem. The challenge is to design efficient switching power supply, which would potentially reduce the power dissipation in the process of conversion (i.e. voltage level boosting).

2.1.1.5 Sensor Node Operating System

A typical example of an operating system used mainly for sensor nodes is the TinyOS [HIL00][LYN05]. TinyOS is an operating system that supports modularity and event based programming paradigms, hence it is suitable for wireless sensor nodes. TinyOS is based on the component based model approach. Therefore, the modularity of TinyOS (reuse of its components) permits it to be the main driver for implementation of different WSN applications.

TinyOS is not like a traditional operating system, but rather a programming structure (with base code less than 400 bytes) for embedded systems (in particular well suited for sensor nodes) with a set of components, which enable development of different applications, and reuse of the existing components.

Some examples of sensor nodes are:

Mica Mote family

Mica2

- *Tiny wireless platform for sensor networks*
- *Uses AA battery (greater than 1 yr. Lifetime when using sleep modes)*
- *Can be used as a router*
- *Has multi-channel radio transceiver*
- *Equipped with expansion connector for external sensors*

Figure 2.2 - Mica2 sensor node

Source: Crossbow (www.xbow.com)

XYZ sensor node

- *Open source wireless sensing platform*
- *Equipped with OKI ML67Q500x ARM THUMB processor and CC2420 Chipcon radio (IEEE 802.15.4 compliant)*
- *Supports different sleep modes*
- *Capable to operate at different speeds and power configurations*

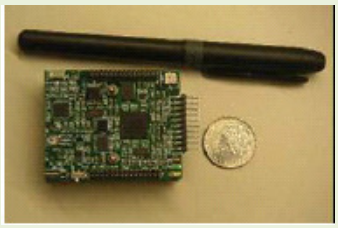


Figure 2.3 - XYZ sensor node

Source:www.eng.yale.edu/enalab/xyz/

Eyes

- *Three year European project on Wireless Sensor Networks (self-organization and energy-efficiency of sensor nodes)*
- *Tiny sensor node platform based on the MSP430 controller*
- *Equipped with RFM10001low power radio module*

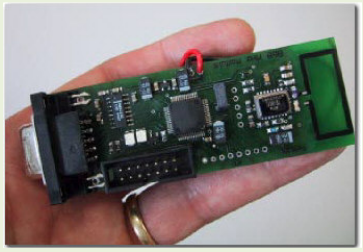


Figure 2.4 - Eyes sensor node

Source: www.eyes.eu.org/sensnet.htm

2.1.2 WSN Topology Control

The objective of WSN Topology Control in general, is related to the efficiency of WSN network in order to increase its lifetime. In a 'nutshell' Topology Control exploits the redundant deployment of the sensor nodes, to overcome the energy limitations by restricting the set of nodes, which are considered neighbors of a given node. While making sure that sensing area is still covered by a sufficient number of sensors. Moreover, reducing the interference problems (i.e., which is noticeable when large number of sensor nodes are active).

The effectiveness of WSN is not just in their monitoring, actuating, computing and communications capabilities: with the added processing power, analog and digital ports, transceivers and memory, they can self-organize and communicate in the deployed area (as depicted in Figure 2.5).

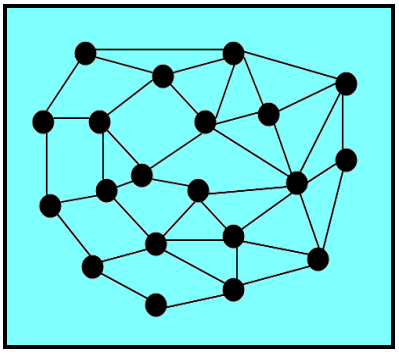


Figure 2.5 - Sensor nodes self-organized

The Topology Control in *flat networks* is mainly concerned with the power control of the sensor nodes – controlling the transmission range and/or the number of nodes' neighbors.

On the other hand, we have the WSN Topology Control in *hierarchical networks by dominating sets and/or clustering;* the main idea here is to choose a set of active nodes within a network, such as to effectively utilize and conserve the nodes' energy by assigning a specific role to the chosen nodes. One example of the Topology Control *in hierarchical networks by clustering* is Low Energy Adaptive Clustering Hierarchy (LEACH) [HEI02]. The assumption here is that the number of nodes and coverage area is known.

From each node's neighborhood (a cluster of sensor nodes) a '*clusterhead'* is chosen. The '*clusterhead'* collects data readings from cluster members and performs data aggregation, prior to transmission of data to the data sink (sensor nodes where the data has to be delivered, typically sinks are more powerful sensor nodes with high transmission power capabilities, more energy resources and computational power). In order to avoid the energy drainage, *'clusterheads'* rotate their role among cluster members. In addition to the above, there are many other strategies that consider the problem of WSN Topology Control, such as the hybrid adaptable approaches, which tend to take advantage of both power and hierarchical control; one such example is The Adaptive Self-Configuring Sensor Networks' Topologies (ASCENT) [CER02]. In the case of ASCENT Topology Control algorithm, the network adapts to the needs of the ongoing communications rather than constructing a 'backbone' or a clustering structure, where each node assesses its connectivity and adept their participation in a multihop network topology based on the operating region.

Sensor nodes that are '*active'* participate in the dissemination of data to the sinks, while the other nodes, which are in '*passive'* state, wake up periodically to check if their participation is needed. Therefore, in the initial state only few sensors are active, taking care of the transmission of data. In the scenarios where the range of transmission from the source to sink (destination node) is at the limit of transmission range, the sink might experience high data packets losses. Consequently, sink issues a *'help message'* to the currently passive neighbors (sensor nodes which are in a listening mode) to join the network. After receiving the *'help message'* node can decide to join the network and announces its change of status by sending a *'neighbor announcement message'* to its neighboring sensor nodes. This process continues until the active sensor node stabilizes, and the network comes to a reliable

operational state (where packet losses from source to sink are minimal e.g. acceptable threshold). During the operation some sensor nodes might fail due to their consumption of energy resources, obstacles or other, where the data packet losses happen again, in which case the above-described process is re-initiated again.

The essence of Topology Control for WSN is to exploit the redundancy of sensor nodes in a network and minimize the number of active nodes, whilst ensuring a reliable and efficient service. In addition, Topology Control helps to extend the life of sensor nodes by compensating for their energy limitations, optimizing the utilization of many sensor operational states. In addition to the above considerations, it is of vital importance to observe and evaluate the behavior of WSN model under different test scenarios prior to adopting any approach. An example of the Topology Control for WSN is provided in Appendix A.

2.1.3 Data Aggregation in WSN

The basic principle of data aggregation can be described as a collective operation of the intermediate nodes performing some form of aggregation function on the data from the neighbouring sensor node e.g. by gathering the data and computing the representation of several messages, which is equivalent or a close approximation of the messages; examples of data aggregation are calculating the average, median, minimum, maximum or other practical values from the measured readings of sensors, prior to forwarding the aggregated value towards a data sink. Therefore, operating on a data whilst is being transported from the sources to the sink. This process is also known as in-network processing.

Main characteristics of data aggregations are:

- ***Accuracy***
    - How well does the value received at the data sink correlates to the true value
- ***Completeness***
    - Percentage of data messages involved in the calculation of the final – aggregated value
- ***Latency***
    - Time element that is involved in the process of data aggregation among nodes could lead to the possible delays in the intermediate nodes

- ***Message overhead***
    - Reduced overhead as a result of a lower number of messages being forwarded throughout a network; data aggregation implies computation of a smaller representation of a large number of data messages at the intermediate nodes within a network

Benefits of data aggregation depend on where the sensors are located with respect to the sink(s). As depicted in the following Figure 2.6 (on the left), the scenario where the sensors are located e.g. in a radial configuration – whereas all the sensor nodes are only hop away from the sink, represent a scenario where data aggregation would not prove to be beneficial. However, in Figure 2.6 (on the right), it can be seen that the data aggregation proves to be beneficial, since the sensor nodes along the network are more than one hop away from the sink, hence the aggregation of the data in the intermediate nodes would lead to less messages overhead through the network, towards a sink. Therefore, data aggregation usefulness depends on the location of the sources of data relative to the sink.

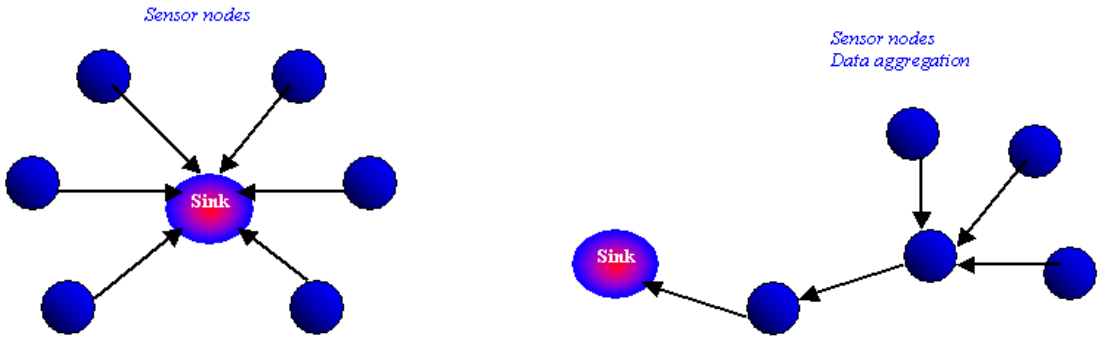


Figure 2.6 - The usefulness of data aggregation in different scenarios

The intermediate result of the aggregated values, typically are representation of the aggregated value and the number of sensor readings that contributed to compute the given aggregated value. In other words, the data aggregation at the intermediate nodes when transported along the network consists of the aggregated value and the count of sensors that contributed in the computed value.

A practical example of data aggregation is Tiny Aggregation (TAG) [MAD02]. TAG is a generic service for aggregation, similar to SQL query languages, which was developed for the ad hoc networks of TinyOS sensor motes. TAG is a popular scheme used for WSN. The basic operation of TAG aggregation scheme is based on the convergecast tree - gathering of data from many or all available sources to a single or several sinks. TAG processes the data aggregation while the data is transported in the network, discarding the redundant and irrelevant data, whilst combining the more important sensor readings into more compact results, when available.

An example of query syntax in TAG would be:

SELECT AVG (Temperature), Room FROM sensors
WHERE floor 5
GROUP BY Room
HAVING Temperature > 25
EPOCH DURATION 45s

The interpretation of the above is the following: The selected parameter of interest is the average temperature in floor 5; sensors are partitioned into group by rooms, query reports when the temperatures are above 25 °C, and the process is repeated every 45 seconds.

2.1.4 Transport Protocol for WSN

The architecture of computer and communications network is usually structured in different layers, such as physical, data, network, transport, presentation, session, and application layer based on the open systems interconnection (OSI) reference model; where each layer provides service to its immediate upper layer as shown in Figure 2.7. The physical layer is the hardware platform of the network responsible for transmission over the physical medium, whereas the data layer provides link services to the network layer. The network layer provides routing and addressing services to the transport layer, and the transport layer provides message transportation services to the layers above it. Transport layer in general provides end-to-end segment transportation, where messages are segmented into a series of segments at the source node, and afterwards are reassembled at the destination node.

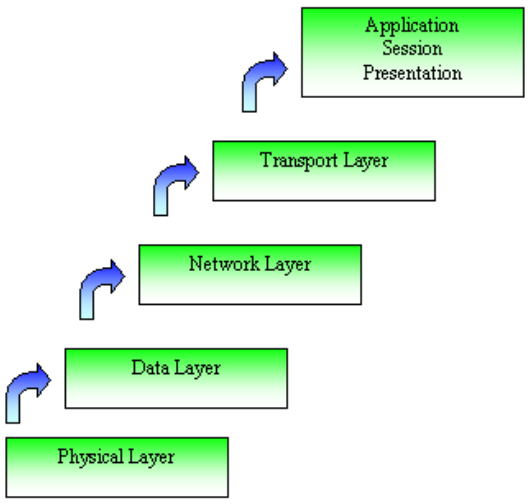


Figure 2.7 - Generic network layering structure

Examples of transport protocols are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). Since there is no interaction between the TCP/UDP and lower layer protocols, TCP/UDP might not be a feasible solution for WSN, where the interaction with lower layers can provide valuable information to the transport protocol, and enhance the overall system performance. In WSN, typically there is a small amount of data to be sensed; hence, the TCP/UDP protocol might be a large overhead. The energy conservation in WSN transport protocol could possibly be related not only to physical layer, but also to data link, network, and other layers. Therefore, the WSN transport protocol design emphasis should be towards the energy conservation and congestion control, loss recovery, security, and management, by taking into account that the energy, memory and computational constraints of sensor nodes.

The aim of congestion control in WSN is to generate less traffic, have limited number of packets being lost, fewer re-transmissions, therefore energy savings in sensor nodes. The congestion in WSN typically happens near the sink (i.e. data transmission is typically from the sensors to the sinks) where Medium Access Control (MAC) can recover the segment loss due to bit errors, however cannot recover the errors due to buffer overflow (i.e. limited memory/processing capabilities of sensor nodes), leading to packets being lost. The packet loss in WSN is mainly due to the congestion, quality of wireless channel or sensor node failure. A practical protocol used for congestion control is *"CODA: Congestion Detection and Avoidance in Sensor Networks"* [WAN03]. The Congestion Control and Avoidance

(CODA) protocol is an upstream congestion control protocol, consisting of congestion detection, open loop hop-by-hop backpressure (from the node where the congestion occurred back to the source) and closed loop end-to-end multi-source regulation. CODA attempts to detect congestion by monitoring the buffer occupancy and wireless channel load. Hence, if buffer occupancy and/or wireless channel load exceed the certain set threshold, implies that the congestion has occurred; where node that has detected the congestion notifies its upstream neighbors (hop-by-hop backpressure process) to reduce their transmission rate, and the neighboring nodes trigger the reduction of their output rate. Although, under normal operation sensor nodes would regulate themselves at predefined rates, when this rate exceeds its theoretical threshold, a source is prone to contribute to congestion hence, it sets a 'regulation bit' in the event packet, and the closed loop control is triggered. This in fact forces the sink to send the ACK signal to all the sources associated with that particular data event. The ACK, which sources expect to receive from the sink, are related to the predefined transmission rate, or to the number of ACK expected to receive over a predefined period. If the source node receives a prescribed number of ACK it maintains its rate. However, since during this process the congestion can build up, ACK can be lost; in which case the sources would reduce their rates independently (e.g. according to some multiplicative decrease function) and the sink stops sending the ACK based on the network conditions (e.g. when the source reporting rates are less than the desired rate). When the congestion is cleared, the sink starts sending the ACK control message to the sources, informing them to increase their rates (e.g. according to some additive increase function).

2.2 Artificial Intelligence

Artificial intelligence is a science with defined objectives of making machines perform things which would require intelligence if performed by humans, which implies capability of performing some forms of cognitive tasking. Thus, here lies the challenge of most intelligent systems today.

One of the major 'paradigm shifts' in AI field was its change of focus from general purpose, weak methods to domain specific methods, which initially lead to the development of expert systems. In addition to the expert systems, AI field was further enriched by introduction of techniques such as fuzzy logic, biologically inspired techniques such as

artificial neural networks (ANN) [NEG05], ART [GRA07], clustering algorithms, evolutionary computations - GA, simulated annealing, PSO, intelligent agents [RUS03] etc; hence, providing vast methods for building intelligent systems with supervised and/or unsupervised learning capabilities [FAU94].

Among vast applications of AI in many facets of our lives, "Intelligent Building and Homes" is an interesting and ever-growing area of research for many scholars, researchers and engineers. Multiagent framework for intelligent building equipped with sensors and actuators, where multiple agents control the parts of the environment using fuzzy logic rules linking sensors and actuators, was described by Rutishauser et al in [RUT05]. However, the proposed system cannot address the stability-plasticity dilemma (i.e. how to learn the new knowledge without destroying the existing knowledge base); as a result the long-term knowledge could be degraded or rendered insufficient for making right decisions if misleading/erroneous samples are introduced. The existing AI techniques such as, ART [GLO76a][GLO76b] could be valuable for scenarios described above, since it resolves the stability-plasticity dilemma (i.e. by clustering together new concepts which are similar to the existing ones, while creating new clusters when encountering new knowledge).

Brdiczka et al in [BRD09] in their research associated to learning situation models for providing context-aware services in a dynamic smart home environment, present intelligible framework consisting of different layers of situation model; the approach taken utilizes expert knowledge, and it consists of several methods which are used to acquire different levels of situation model (i.e., role detection, unsupervised extraction of data, and supervised learning, and integration of user feedback/preferences).

The conducted experiments utilize multiple tracking cameras, microphones, and sensors having to be installed in room, including many processors for processing and analyzing video and audio streams. Incorporating occupancy sensors with capacitive sensing [GEO09] might prove efficient with respect to price and processing power for scenarios such as, sitting and/or lying down. The feasibility of intelligent systems lies in its ability to learn and adapt with minimal required interaction by the user during its operation. A vital role and purpose of AI techniques (among others) is to reduce the complexity of hardware by introducing low computational overhead.

2.2.1 Expert Systems

The rule-based and/or frame-based expert systems are a typical approach taken to represent and build knowledge-based systems [SHO76][DUD79][NEG05][WAN10]. The rule-based expert system uses if-then rules, while the frame-based expert system uses 'frames' (i.e. objects and/or structures) to represent the knowledge.

A rule-based expert system consists of if-then rules (conditions and actions) and can have multiple conditions to represent the knowledge needed to solve a problem in a particular domain of study.

If $x_1, x_2 \ldots x_n$ represent the conditions for a particular problem; and $y_1, y_2 \ldots y_n$ represent the actions to be taken if a particular condition(s) are true, in a rule-based expert system, rules can be expressed as shown in examples below:

A simple if-then rule:
if (x_1)
then (y_1)

A rule with multiple conditions:
if (x_1 **AND** x_2 **AND** $x_3 \ldots$**AND** x_n)
then (y_2)

A rule with multiple mixed (AND/OR) conditions:
if (x_1 **AND** x_2 **OR** x_3 **OR** x_4 … **AND** x_n)
then (y_n)

The main idea behind rules as a knowledge representation technique is to reflect the knowledge of an expert in a specialized narrow domain, for which the system is being developed. Depicted in Figure 2.8 is a conceptual model of typical rule-based and/or frame-based expert systems.

The knowledge-base contains the domain knowledge as a collection of if-then rules, which in conjunction with facts (database of facts) is used as a 'filtering criteria' of the

reasoning process, by linking the rules with the facts and inferring the conclusion (i.e. solution of a particular problem and/or user inquiry).

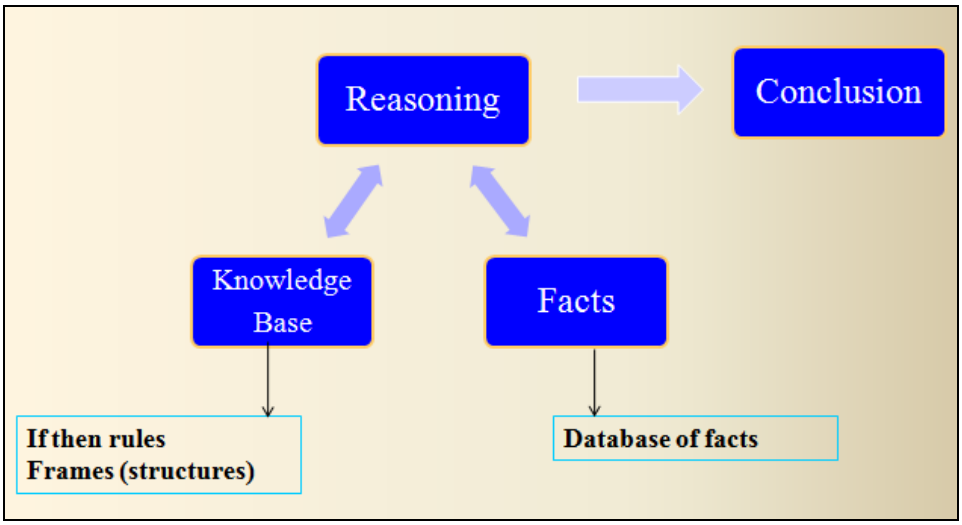


Figure 2.8 - Generic conceptual model used in rules-based(frame-based) expert system

There are two fundamental principles based on which the rules are executed: the forward and backward chaining. The forward chaining is based on data driven reasoning, where the reasoning starts with the known data, and the rules are fired only if there is a match of facts in the database (of facts). And it continues forward each time (i.e., when a rule is fired) adding new facts to the database. This is a technique which first gathers the data, and afterwards infers from it. Whilst the backward chaining instead is a goal driven reasoning, where the expert system starts with a hypothetical solution or a goal, and the aim is to prove it. The inference engine in backward chaining does not fire the rules based on the known facts (as is the case of forward chaining) instead, it checks to find the rule that matches the hypothetical solution or the set goal. If the solution is not found, it sets up a new sub-goal and continues by checking if the sub-goal can be proven based on the existing database of facts. The process continues until the goal is found and/or there are no matching rules to fire (throughout the search process).

The frame-based expert systems on the other hand, offer several advantages which are not available in rule-based expert systems, such as, using data structures in order to encompass the necessary knowledge about a particular object and/or concept [WAN10] [RAT07] instead of simple if-then rules. Frames are described by a collection of attributes

and characteristics of objects of interest, which are called slots. In addition, frames consist also of facets which are value based (extended knowledge about a particular attribute) and prompt (user) based facets, which allow user to enter attribute values for a particular object of interest. The instance-frames are distinct-particular objects and the class-frames are a group of similar objects. The concepts of object oriented programming such as the inheritance and also methods associated with objects and/or classes are adopted also in frame-based expert systems.

In addition to the rule-based and frame-based expert systems [ALT85] [KAM94] [WEN10], the so-called 'expert system shells' nowadays are also very useful [PAM10]; they enable the researchers to concentrate on the knowledge-base representation instead of programming language. Although being able to understand and interpret both can be beneficial during the implementation and development stages.

2.2.2 Intelligent Agents

In the context of AI and AmI, an agent is a device that senses the environment via sensors and acts upon sensory inputs via actuators. The added features of an agent could be its capabilities to make intelligent decision about certain operation(s) to be performed, leading to intelligent agent concept [RUS03]. The agent function defines the actions of an intelligent agent related to sensory data inputs, whilst the agent program is its very implementation. An agent program is a tangible implementation of code for i.e., wireless sensor/actuator node which percepts via sensor, executes a matching rule, and takes action via actuator, as depicted in Figure 2.9. There are several different agent program types, such as simple reflex agents, model-based reflex agents, goal-based agents and utility-based agents [YIG09] [SCH07][SAK93][WU08][BOG06][HAR10][CUN08][SIL08]. Simple reflex agents are the simplest form of intelligent agents, which do not take into account the percept history during their actions, and only consider the sensory input being sensed. This is mainly done via matching of the sensory input with the rule matching function. The model-based reflex agents, on the other hand, use an internal model to keep track of the current state of the environment around the agent; it also includes additional state(s) which assists the agent to understand the consequence of actions (if taken).

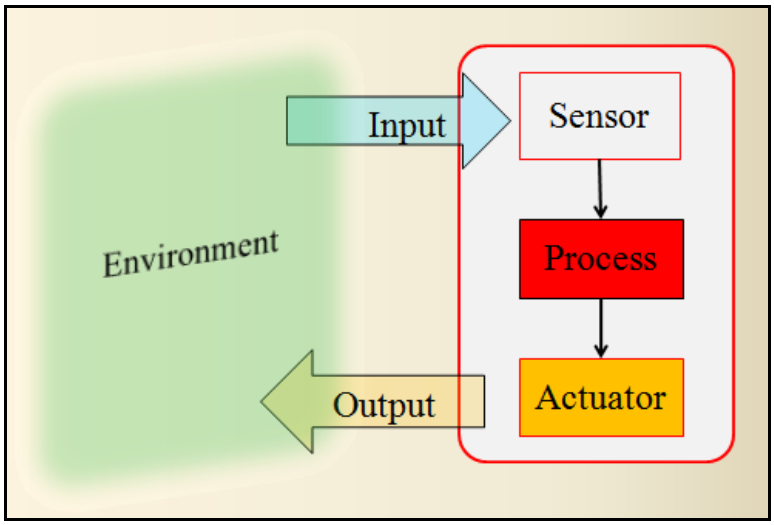


Figure 2.9 - Agent interacting with the environment

The goal-based agent, in addition to the capabilities of model-based reflex agents, includes information about a set of goals which an agent is trying to achieve; which, aids in the decision process of an agent related to the desired state and/or action to be taken. However, considering the goal as the only option for the high quality solution might not always be the best choice, especially for scenarios when more than one solution exists (i.e. due to the complexity of the environment, etc.). A distinctive attribute of the utility-based agents is utility function - a measure which describes the performance element i.e. 'degree of happiness' associated with a certain action being taken. In other words, utility function introduces the means to measure the likelihood of success against the importance of a certain goal.

The above mentioned agent programs are the 'essential ingredients' necessary to create a learning agent and multi-agent learning systems [BI04][REN03][GEL01][JAV10]. The learning element is one essential component of the learning agent, which is responsible for potential agent's improvements, such as to perform better in the future. This is achieved via a feedback element of the learning agent which decides how well the agent is performing based on some defined criteria. The learning agent, encompass also the exploration of environment as an important element of learning process and finding potential alternative solutions.

2.2.3 Reinforcement Learning

Reinforcement learning (RL) is a widely used learning technique for agent and multi-agent based systems. An agent-based model without feedback could have a limited set of capabilities to learn and adapt in an open environment. The feedback mechanism of an agent interacting with the environment provides a 'good ground' to learn optimal 'agent's action policies,' which can be evaluated by means of reward and punishment; assisting into 'awareness' of agent about success and failure rates of its actions via feedback (i.e., reinforced learning), which leads to an improvement of agent performance. However, it has to be noted that the agent's action policies can lead to local instead of global optimum; hence, in addition to the choice of optimal parameters of the reinforcement learning algorithms, cooperation and coordination in multi-agent systems during the learning process can be more effective means to achieve better results [JU04][BER00][ARA00].

One of the limitations of RL techniques is the required memory of a system to store possible states and actions when an agent operates in a dynamic environment, such as for example, as described by Yen et al.[YEN02] tradeoff between exploration and exploitation of the environment are very important factors to be considered; in problems involving large number of states and/or actions, learning process of an agent might turn out to be very slow, and on the other hand, storing all possible states could take vast amounts of memory.

Amongst other AI techniques, RL has a vital role in the emergence of autonomous agents and/or robots in real environments and vast range of applications and or variations of RL approaches in machine learning [JIN10][YAM96][HES10]. RL techniques have the capabilities to learn from experience, whilst approximate reasoning technique lacks this attribute, however provide simple yet powerful means for knowledge representation; when both schemes are combined can lead to efficient hybrid systems used for control (where RL is used for fine-tuning of fuzzy logic rules) [BER91].

The RL techniques are also used in context aware systems [TUD10], whereas a self-adapting algorithm characterized by four phases, uses the reinforcement learning technique in its planning phase, in order to explore the possible system's states and for the selection of actions to be executed by system in case of context changes.

2.2.4 Adaptive Resonance Theory

ART is an unsupervised learning algorithm with biological motivations. ART1 [CAR87a] network model is a self-organizing architecture, designed for clustering binary vectors. ART1 works with objects called feature vectors, which are a collection of binary values representing information, which can be separated into different clusters. Typically, new clusters are created when new data are encountered, whereas the vigilance parameter ($0 \leq \rho < 1$) is used to determine where to place new data, based on the threshold (i.e. to determine the cluster size). The vigilance parameter enables one to control the degree of similarity of data which can be placed on a same cluster (i.e. set of similar data). ART efficiently solves the problem of stability/plasticity dilemma by learning without destroying existing knowledge (i.e. existing cluster are not altered). The basic architecture of ART network is depicted in Figure 2.10.

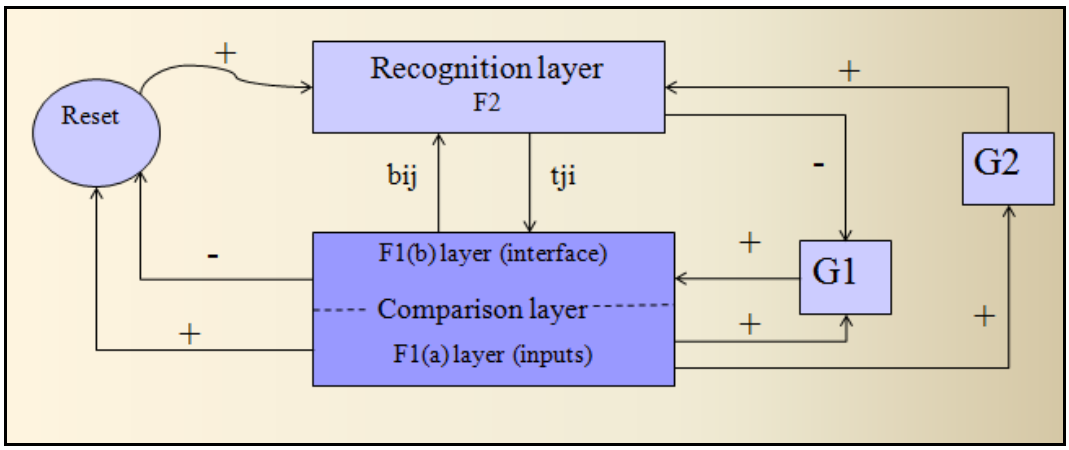


Figure 2.10 - ART network architecture

Where,

F1 layer: Comparison layer (short term memory)

F2 layer: Recognition layer (long term memory)

ρ: Vigilance parameter ($0 \leq \rho < 1$) (controls the cluster size)

b_{ij}: Bottom-up weight vectors

t_{ij}: Top-down weight vectors

The comparison layer F1 processes the input patterns, whilst the recognition layer F2 classifies the cluster units based on similarities. The F1 layer is connected to F2 layer via bottom-up weight vector b_{ij}, and the F2 layer is connected to F1 layer via top-down weight vector t_{ij}. Similarity of top-down weight vector with the input vector is a factor which decides whether a particular cluster unit is allowed to learn or not. The G1 and G2 are gain control units, used to aid the reset mechanism to disable or enable competing units to fire.

A variety of different ART implementations have been demonstrated since the introduction of ART1; such as ART2 [CAR87b] and Fuzzy ART [CAR91a] models, which were designed to work with analog and/or binary input patterns. Furthermore, ART1 was also used to design ARTMAP [CAR91b] in order to be able to learn arbitrary mapping of binary input and output patterns.

The ART network models applications include and are not limited to pattern and image recognition, engineering and manufacturing, remote sensing, medical diagnosis, robot behavior learning and vision [FUN03] [LI10] [SMI94][AMI10][BRA09].

2.3 Ambient Intelligence

Ambient Intelligence (AmI) is a new multidisciplinary paradigm, a concept and vision of future technologies embedded in our living environment, providing support and assistance while preserving security and privacy of the users.

2.3.1 Overview of Ambient Intelligence

Vision of AmI and ubiquitous computing, involves integration of sensing/actuating "intelligent agents" into our living environment, where integration and communication among different types of these smart devices is essential [WEB05]. From the technical point of view, AmI utilizes the existing technologies and intelligent techniques in order to orchestrate the distribution of electronic intelligence - embedded in our living environment. Today, there are no established standards that cover all the aspect of AmI concepts. However, there is a consistent and active research being done in this new and interesting area. The concepts such as the multi-agent sensors and AI techniques are being considered for integration within the AmI concepts.

The development of AmI applications that adapt to the user preferences and environment, among others, requires a well defined architecture and a planning mechanism for a goal-oriented behavior [AMI05]. This aspect of AmI has not been fully explored nor addressed extensively, especially the aspect of coordination among devices and application of distributed intelligence, in order to provide a more versatility that can be applicable for different applications and use.

The lack of a general reference models for AmI system designs, has led many researchers to investigate and bring forth hierarchical conceptual model for AmI architectures [RUI09], where versatility and applicability is essential element of system design space; while the importance of network and middle-ware layer (intelligent kernel) is considered as "the brain" of the AmI space. In addition, multi-agent model and interaction of different agents and challenges of heterogeneous data exchange among agents is an important aspect considered in AmI space design. However, lack of self-organization is an additional aspect which has to be considered carefully for scalable and flexible AmI solutions, since one of the main objectives of AmI applications should encompass design characteristics, such as autonomous, adaptive and learning systems. Thus, a challenge remaining for the AmI to evolve - is the necessity to expand its boundaries, by embracing more critically AI techniques.

2.3.2 Applications of Artificial Intelligence Techniques in Ambient Intelligence

AI techniques are essential for AmI to become a successful multidisciplinary model [REM05] and perhaps 'The next step in AI's evolution' [RAM08]. In order for AmI to reflect adaptability, anticipation and learn the user needs and patterns, it requires some form of intelligence. Applications of AI techniques related to machine learning, intelligent agents and robotics complement the AmI vision of a smart environment - capable of sensing, interpreting and representing the information, learning about the environment, anticipating, perform variety of tasks, and interact with humans in a non-intrusive and user-friendly manner [AUG07a][COO09][LOC10].

From the AmI enabling technologies and main architectural blocks of AmI [INF01][RAM08], the 'Knowledge and Reasoning Layer' is where the contribution of AI is noticeable [HOO10], where the application of techniques such as knowledge representation,

information retrieval, expert systems, computational intelligence, multi-agent systems, etc. are an integral part of AmI overall system architecture.

Application of AI techniques in AmI context, encompassing not only intelligent system design, but also consider the user friendly interfaces, efficient services support, user-empowerment and support for human interaction in a context of an intelligent transport system are discussed in [MIL06]. Furthermore, embedded intelligent agents [HAG04] are utilized in iDorm in order to add 'the intelligence factor' to the AmI, by aiding in reasoning, learning and planning aspect of the overall system; hence, assisting in building a vision of AmI.

In [JIA08] Jian et al] propose Multi-Agent System based architecture of the AmI system. Whereas, multiple intelligent agents are distributed in different appliances and react and reflect autonomy and cooperates with other agents in order to provide personalized and automated services to the user. Whilst, Augusto and McCullagh in [AUG07b] describe the concepts and applications of AmI, in particular the relationship between AmI and related areas, such as human computer interaction, sensors, networks, ubiquitous computing and artificial intelligence; whereas the AI reasoning, knowledge repository, discovery learning and decision making, fit into the architecture of a typical AmI system.

CHAPTER 3

ADAPTIVE SYSTEMS FOR SMART BUILDINGS UTILIZING WSN AND AI

3.1 Methodological Approach

In order to provide adaptive systemic solutions for our living environments, one should observe and analyze the problem from a 'systems perspective' and not just focus on the sole problem of how to make a PCT smarter. The emphasis of this research is to utilize multiple variables at hand (i.e., WSN agents and AI adaptive learning techniques) in order to bring forward an adaptive system for PCTs. By introducing multiple variables, albeit it might seem that we are adding complexity to the problem. However, by applying the principles of system interaction (i.e., utilizing the diversity of resources: WSN, AI and AmI) we facilitate and simplify the overall control, and the learning process of the system itself. The principles of system interaction are in fact analogous to interdisciplinary research concepts, where many fields of science work together to acquire a needed solution to multi-dimensional problems. Similar principles can be applied to any problem which encompasses many facets of discovery; in this particular case, development of an *Adaptive System for Smart Buildings Utilizing WSN and AI.*

The envisioned system mentioned above, consists of few subsystems sharing knowledge and data to achieve a better outcome (i.e. considering that a system is a collection of several subsystems). The WSN agents and AI based techniques enable the system to interact with a multitude of sensor data and to use the existing knowledge base. Furthermore, by exploiting the rule-based expert system and adaptive learning principles, the system is capable of learning and adapting, by using existing knowledge, and creating new knowledge, as well.

The rule-based expert systems cannot offer the best possible solution alone. They have a drawback, which is their dependence on prior knowledge and also the amount of time required to match a rule with the memory. Thus, using only one AI technique in this case would not render a flexible solution. Therefore, the goal is to provide a flexible adaptation

of the overall system to new knowledge without destroying the existing knowledge, i.e., an adaptive learning system for energy conservation and comfort zone adjustment. Since our main objective is to create an adaptive learning system, existing models, such as ART, expert systems, and other AI techniques are investigated to determine the best possible choice, for the system being proposed and considered in this research. One critical element of this research is to combine several AI techniques, and explore innovative solutions, leading to an *Adaptive Learning System* (ALS) - a hybrid intelligent system capable of adapting and learning in our living environments.

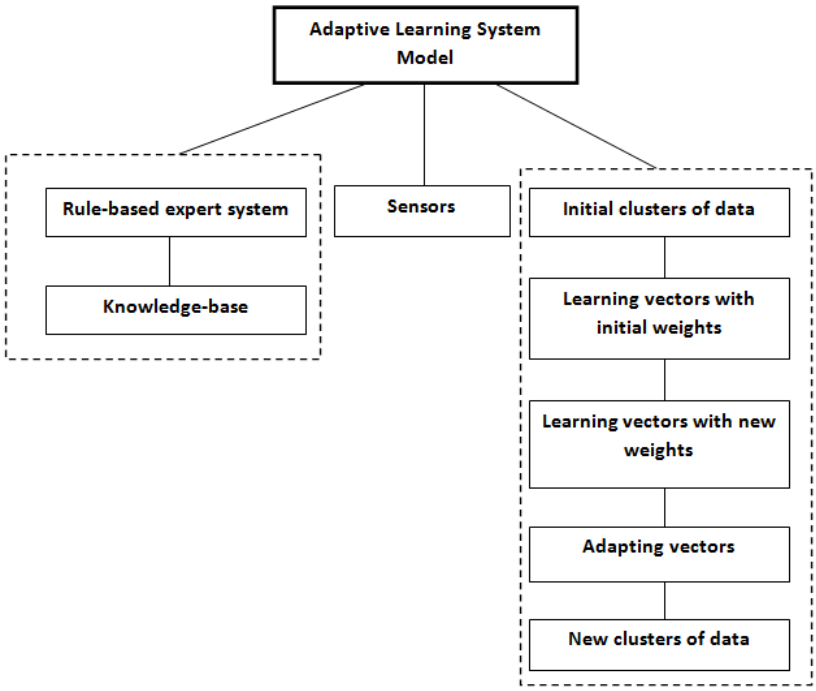


Figure 3.1 – ALS conceptual block diagram

The ALS conceptual block diagram shown in Figure 3.1 depicts the main buildings blocks of the system. The rule-based expert system provides the decision rules, and is used to compare the existing knowledge with the new knowledge being introduced. The Knowledge-base contains information about the heating/cooling stages of the HVAC system being used, available zones to control, air flow rates for different temperature ranges, thermal characteristics of the house, etc.

The environment is constantly monitored via sensors which are capable of detecting the temperature changes, airflow, and the activity/presence of occupants. Thus, allowing the

ALS system to gather input data, and detect when the occupant's schedules or patterns are changing. The actuators enable control of the air dampers within different zones of the environment being conditioned, and the heater/cooler stages of the HVAC being controlled.

The learning vectors learn preferences and patterns based on the user input and sensors (i.e. which is reflected in the weight factor of each element), while the adapting cluster vectors extract information from the learn vectors and adapt to the changes, when new patterns (preferences) are detected. The term cluster in the ALS model refers to an initial set of clustered data (i.e., group of data, daily clusters), which in our case are: PCT daily schedules consisting of temperature set points and their associated time intervals (based on the occupant's preferences) for each day of the week, DR and TOU price incentives, offset temperature tolerances, number of zones to control, etc. In addition to daily clusters, a master cluster maintains the record of each active daily clusters (refer to the Appendix C for further information on the Daily and Master structures used in the implementation of ALS model via OLA algorithm).

On the other hand, the learning vectors hold the occupant's temperature set points and offset tolerances for different times of the day, preferences and the associated weights, indicating if the element's value has changed or not (i.e., learning vectors hold information for each daily cluster). The learning vectors are updated based on the activity being detected by the sensors; hence, changes related to each particular element, are recorded in the adapting vectors. Based on the changes being detected, the initial weights associated with each particular element are updated. Since our objective is to adapt to pattern changes, rules-based expert system is used to analyze and decide (i.e., based on the weights associated with each element, and the existing knowledge). Hence, if the pattern and/or preference change of a user is persistent (i.e., not just a onetime occurrence) it is considered for adaptation. The weights created based on the user preferred tolerances (for any parameter of interest) are used to determine if the change is considered for adaptation or not. Thus, the detected values are applied in order to update the existing elements with the new values. Thus, the adapting cluster vectors are created. Based on the number of occurrences (i.e. the number of occurrences after which the adaptation takes place), the adapting cluster vectors values are compared. If the values are within the scope of the occupant's set limits and/or tolerances, a new cluster is created. Thus, offering a new clustered knowledge (i.e. without

destroying the existing knowledge) based on the occupant's preferences, pattern and/or schedule changes.

This PhD thesis roadmap and approach taken underline the importance of making a scalable solution, starting with only a few inputs / outputs and simple UI, and emerging into a more complex system with multiple sensors, sources of information, and a variety of output types. The solution extends beyond a PCT by proposing an adaptive – hybrid intelligent system, which does not require constant programming input by the occupant, a system that learns and adapts (while offering optimal comfort and energy conservation). The system communicates with the EGU Smart Meter to provide the most efficient savings (TOU price rates), and to help manage the peak load demand (DR incentives). The aim is to create a system which can optimize the comfort with respect to energy consumption by learning the occupancy preferences, schedule changes and patterns; enabling energy savings and comfort zone adjustment of the environment.

3.2 Analytical Modeling

The ALS model proposed therein utilizes WSN and AI concepts from a rule-based expert system and ART, in order to bring forward a novel ALS technique, as described in detail below. Moreover, the ALS model could potentially find its use in other applications, which require some form of adaptive learning and/or intelligence, in addition to its sensing capabilities, where the system learns and adapts based on the occupant's schedule, preferences, and/or pattern changes. The main objective of the ALS is to adapt to the occupant's pattern and/or schedule changes by providing comfort, while not ignoring the energy conservation aspect. It has to be noted that the ALS model ensures that the existing knowledge is not destroyed by the new knowledge (i.e. by pattern and/or schedule changes of the occupant). Thus, the ALS verifies if the new knowledge is already available, and if not, it is added to the existing knowledge-base as new knowledge. A simplified flowchart showing the main concepts of the ALS model is depicted in Figure 3.2, while further details are provided in the following subsection 3.2.1 "Adaptive Learning System (ALS) Model".

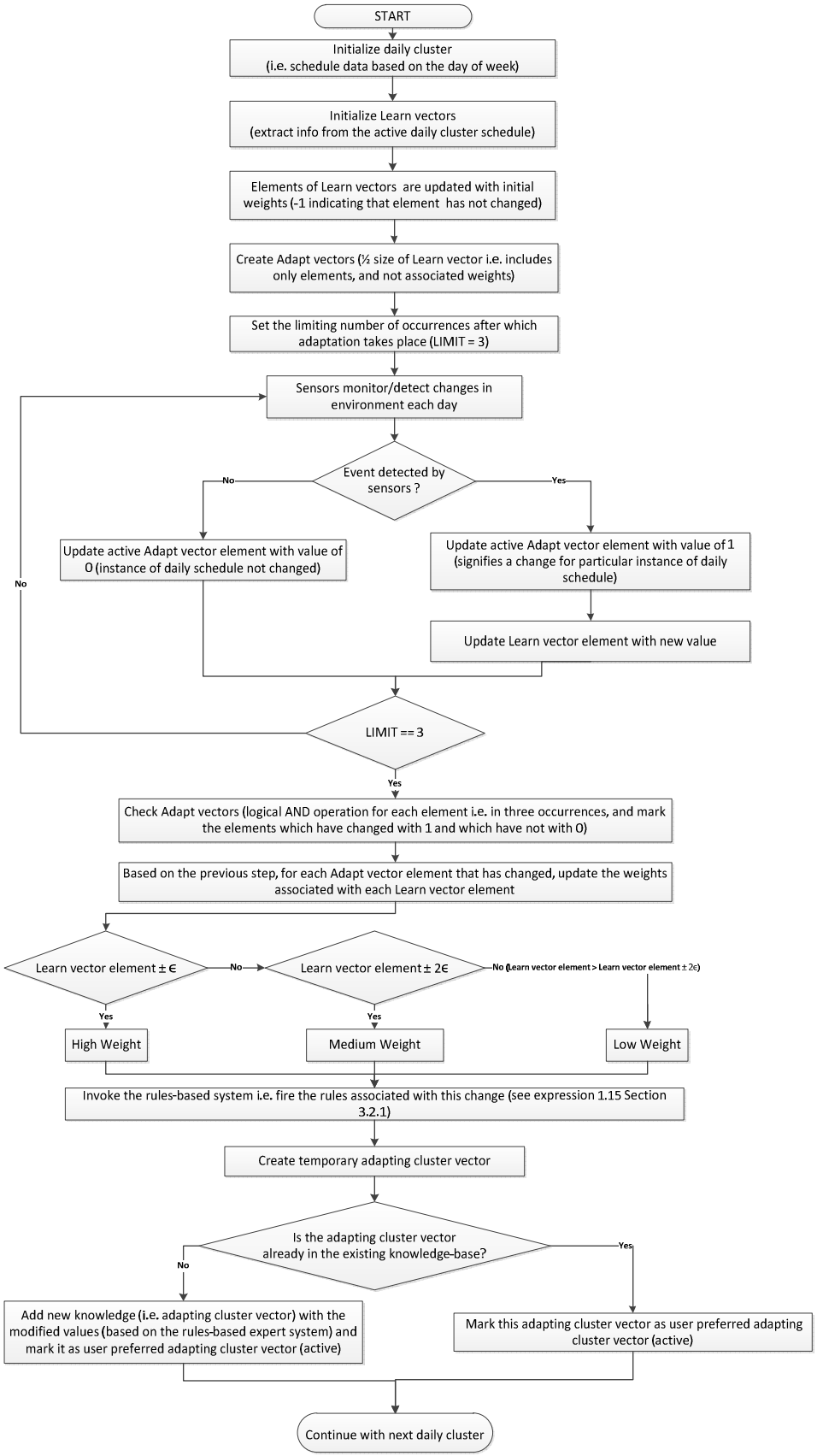


Figure 3.2 – Basic ALS model flowchart

3.2.1 Adaptive Learning System (ALS) Model

Let m represent the number of wireless sensors: s_1, s_2,…, s_m available in a smart environment, and let g = 1,2, …, R represent the readings of each sensor group, for every week day, such as:

$$S_g = \{S_1, S_2,\ldots, S_R\} . \quad (1.1)$$

The sensor groups S_g represent active wireless sensor readings in different locations (or rooms) within a smart environment.

Let *L* represent a learning vector with the following elements, each associated with a weight placeholder:

$$L = \{l_1, l_2,\ldots, l_N, w_{N+1}, w_{N+2},\ldots, w_{N+N}\} . \quad (1.2)$$

The l_1 to l_N elements of the learning vector represent the actual values of interest, such as heat and/or cool set points, set point start and end times, user preferences, etc. The weights are associated with each element of the learning vector i.e., w_{N+1} is the weight associated with the learning element l_1, w_{N+2} is the weight associated with the learning element l_2, etc.

Let A_c represent an adapting vector:

$$A_c = \{a_1, a_2,\ldots, a_N\} . \quad (1.3)$$

The vector A_c includes only the values of interest (and not the associated weights). Thus, for each elements l_1 to l_N of the learning vector, there is a corresponding element of the adapting vector a_1 to a_N. Furthermore, let $\boldsymbol{c_{ij}}$ represent all the existing clusters under consideration, where i = 1,2, …, 7 represent the number of weekdays under observation, and j = 1, 2,…, N number of occurrences for every event under observation, for each weekday. In addition, let *A** represent a set of the corresponding adapting cluster vectors:

$$A^* = \{A_{ci1}, A_{ci2},\ldots, A_{ciN}\} . \quad (1.4)$$

The vectors A_{c11} to A_{c1N} represent a set of adapting vectors (daily cluster) under observation i.e., for every event under observation, for each weekday occurrences.

And let L^* represent a set of learning vectors under observation, where:

$$L^* = \{L_1, L_2, \ldots, L_N\} \ . \qquad (1.5)$$

Where L^* is a set of learning (cluster) vectors L_i, $\forall$ i = 1, 2,…, N, where N is the maximum number of learning occurrences (i.e., for each weekday). Furthermore, for every learning vector L_i, for a weight $\boldsymbol{w}_{N+k}$, initial weight conditions are:

$$\boldsymbol{w}_{N+k} = -1 \ . \qquad (1.6)$$

Where constant N is a number representing the size of learning vector elements, and k = 1, 2, …, N represents the corresponding weights associated with learning vector elements. From the Equation 1.6, it has to be noted that the assigned value of -1 is the initial default value of the weight $\boldsymbol{w}_k$, which signifies that the value of learning vector element corresponding to its associated weight has not changed; whilst, any other value indicates a change of element's value.

Expression 1.7 is used to perform the conditional checks, in order to initially populate the cluster vectors. For each learning vector element, where k = 1, 2,…, N (adapting vector is half of the size of learning vector, i.e., it does not include the weights):

$$\begin{aligned} &\textbf{\textit{if}}\ (L_i[\mathrm{N+k}] == -1) \qquad (1.7)\\ &\quad A_{cij}[\mathrm{k}] = 0\\ &\textbf{\textit{else}}\\ &\quad A_{cij}[\mathrm{k}] = 1 \end{aligned}$$

The A_{cij}[k] assigned value of 0 signifies that the value of the element at index k did not change, whilst the value of 1 means it has changed. It has to be noted that the above expression (1.7) is repeated for every existing cluster (the occurrences of patterns within a reasonable interval of interest).

Now, after the initialization of all the adapting vectors with default values, according to the number of occurrences considered and detected by sensors S_i, the conditional expression 1.8 is necessary, in order to record if any of the elements were changed.

Let T be a temporary vector and A_{cij} vectors to adapt (i.e., new vectors of a specific cluster indicating the state of elements needed to be modified within each cluster, due to the occurring changes).

$$\begin{aligned} &\textbf{\textit{if}}(A_{c11}[k]== 1 \text{ AND } Ac_{12}[k]== 1 \text{ AND } \dots \text{ AND } A_{c1N}[k]== 1) \qquad (1.8)\\ &\quad T[k] = 1\\ &\textbf{\textit{else}}\\ &\quad T[k] = 0 \end{aligned}$$

After which the values of T are assigned to the A_{c1j} vectors to adapt, for each element.

$$A_{c1j}[k] = T[k] \; . \qquad (1.9)$$

The same procedure shown in (1.8) and (1.9) is repeated for all the vectors to adapt A_{cij}, for each existing cluster. Where, c_{1j}, c_{2j},…, c_{Nj}, represent different clusters. And j represents the number of occurrences within a particular cluster data.

The new data extracted from the sensor sets i.e. S_g are used by learning vector L, for every existing cluster (i.e. daily cluster) in order to compare the currently sensed values with the previous ones; hence, detect if a change is obvious or not, for each element.

$$\begin{aligned} &\forall \; i = 1, 2 \dots N \text{ and } \forall \; k = N+1, N+2, \dots, N+N\\ &L_1[i] \leftarrow S_1[i], L_2[i] \leftarrow S_2[i] \dots L_N[i] \leftarrow S_R[i] \; . \qquad (1.10)\\ &L_1[k] = -1, L_2[k] = -1, \dots L_N[k] = -1 \; . \end{aligned}$$

Where S_1, S_2,…S_R are the sets of wireless sensor readings of each sensor group S_g, for each day of a week; representing the sets of sensor readings for different instances of interest (i.e., different days of a week, different patterns and/or scenarios under observation).

The values, 1 to N - represent the number of elements in a learning vector, and the values N+1 to N+N - represent the number of the elements' associated weights. Note, that the weights associated with elements initially are populated (initialized) as -1 indicating that yet

no change has occurred. Thus, prior to starting learning process, each weight is assigned the value of -1.

Now, based on the cluster data available in learning vectors L, the L_{cij} learning vectors include each particular daily cluster changes (each representing a different vector corresponding to a specific occurrence), as shown in the following equation:

$$L_{c11}[k_1] = L_1[k_1]. \quad (1.11)$$
$$L_{c12}[k_2] = L_1[k_2].$$
$$\dots$$
$$L_{c1N}[k_n] = L_1[k_n].$$

Where,

k_1 – represents the number of elements in the 1st set under evaluation

k_2 – represents the number of elements in the 2nd set under evaluation

…

k_n – represents the number of elements in the nth set under evaluation

The same equation is applied also for the L_{c2k}, L_{c3k},…, L_{cNk} clusters.

Let ϵ represent different tolerances based on which the particular elements' weights are evaluated, and the decision with respect to adaptation is considered.

Where,

$$\epsilon_{low} < \epsilon_{med} < \epsilon_{high} \quad (1.12)$$

Let w_{low}, w_{med}, w_{high} represent different weights associated with the elements of clusters, decided by the expert system based on the application considered. The following expression (1.13) ensures that the weights for different elements are assigned according to their values chosen for a particular problem at hand. In this case, for each element of the learning vector if the corresponding weight value is not -1, it indicates that a change was observed, hence based on the sensed value and its tolerances with respect to typical desired value, new weight is assigned, corresponding to that particular element of the learning vector.

$$
\begin{aligned}
&\forall\, i = 1, 2, \ldots, N \\
&\textbf{if } (L_{c11}[N+i] \neq -1 \text{ AND } L_1[N+i] \neq -1 \text{) Then} \\
&\quad \textbf{if } (L_{c11}[i] - \epsilon_{low} \leq L_1[i] \leq L_{c11}[i] + \epsilon_{low}) \\
&\qquad L_{c11}[N+i] = w_{high}\,. \\
&\quad \textbf{if } (L_{c11}[i] - \epsilon_{med} \leq L_1[i] \leq L_{c11}[i] + \epsilon_{med}) \\
&\qquad L_{c11}[N+i] = w_{med}\,. \\
&\quad \textbf{if } (L_{c11}[i] - \epsilon_{high} \leq L_1[i] \leq L_{c11}[i] + \epsilon_{high}) \\
&\qquad L_{c11}[N+i] = w_{low}\,.
\end{aligned}
\tag{1.13}
$$

The above expression (1.13) is repeated for all existing elements L_{c1k}, L_{c2k}, L_{c3k},…, L_{cNk} of clusters under observation (i.e., daily cluster occurrences) with respect to learning vectors: L_1, L_2,…, L_N. Thus, it establishes the adaptive approximation between the values and weights, based on the resulting sensor inputs (values received) and set tolerances.

Following the weights associated with each element of cluster vectors L_{cik} for each particular cluster, another conditional check is performed. The weight checking process is the final check where previous steps (i.e. weights assigned based on data tolerances ϵ) are used, and the rule-based knowledge is used in the decision making process, in order to adapt the new knowledge within the existing clusters. Based on the weights assigned i.e. w_{low}, w_{med}, w_{high}, the possible outcomes related to the adaptation, for each element of every cluster considered are used (i.e. the following approach, can be modified for use in different applications with similar requirements):

λ - is the learning rate, where: $0 \leq \lambda \leq 1$

wl_{ow}, w_{med}, w_{high} - are the weights associated with each element of the vector under consideration, and does not necessarily have to be only three weights. Number of weights can be chosen based on the complexity and granularity of the adapted values under consideration.

β - is the weight multiplier coefficient, which can be adapted, based on the problem at hand (typically, $\beta = 1$).

Therefore, for the problem under consideration, which consists of three important weights, noted as w_{high}, w_{med} and w_{low}, we have:

$$w_{high} = \beta \lambda .$$
$$w_{med} = (\beta/2) \lambda . \quad (1.14)$$
$$w_{low} = (\beta/4) \lambda .$$

Furthermore, for simplicity, we consider the case of three different clusters, for learn and adapt process.

Let k = 1, 2, 3 and cluster vectors under consideration L_{c1k}, L_{c2k}, L_{c3k}. Hence, the following clusters are considered for adaptation:

L_{c11}, L_{c12}, L_{c13} - are three occurrences of cluster one,

L_{c21}, L_{c22}, L_{c23} - are three occurrences of cluster two, and

L_{c31}, L_{c32}, L_{c33} - are three occurrences of cluster three, while

$\Theta[k]$ - are the resulting values of the adapted elements.

There are three different weights: w_{high}, w_{med} and wl_{ow}, which can be assigned to any daily vector based on the proximity of the actual value to the particular element's value (i.e. three consecutive occurences of a particular day). The rules-based decision according to the weights, is based on the following possible combinations: LLL, LLH, LHL, LHH, HLL, HLH, HHL, HHH, MMM, MMH, MHM, MHH, HMM, HMH, HHM, LLM, LML, LMM, MLL, MLM, MML, LMH, LHM, MLH, MHL, HML, HLM. Where, H stands for w_{high}, M for w_{med} and L for w_{low}.

The resulting values returned from the rules-based decisions are based on the weight occurences i.e. in cases when all three weekly occurences have the same weights, average of the weekly elements is returned. Otherwise, if only two weekly occurences of any particular elements have high weights, while the third one has low weight, the resulting value returned is based on the average of the data elements corresponding to high weights, ignoring the low weight element. Furthermore, the low weight signifies a major shift from a typical existing value, hence the approach is slightly conservative and tends not to make radical changes to the existing schedule or set points. Hence, in cases, such as when major shifts from the existing schedules and/or setpoints occur, adaptation will take place only after three consecutive occurences of the low weights.

Rule-based decisions (1.15) according to the weights (high, medium and low) are depicted below, $\forall$ i = 1, 2,…, N:

$$
\begin{aligned}
&\textbf{if } (L_{c11}[N+i] == w_{high} \; AND \; L_{c12}[N+i] == w_{high} \; AND \; L_{c13}[N+i] == w_{high}) \\
&\Theta[i] = (L_{c11}[i] + L_{c12}[i] + L_{c13}[i])/3 \\
&\textbf{if } (L_{c11}[N+i] == w_{low} \; AND \; L_{c12}[N+i] == w_{high} \; AND \; L_{c13}[N+i] == w_{high}) \\
&\Theta[i] = (L_{c12}[i] + L_{c13}[i])/2 \\
&\textbf{if } (L_{c11}[N+i] == w_{high} \; AND \; L_{c12}[N+i] == w_{low} \; AND \; L_{c13}[N+i] == w_{high}) \\
&\Theta[i] = (L_{c11}[i] + L_{c13}[i])/2 \\
&\textbf{if } (L_{c11}[N+i] == w_{high} \; AND \; L_{c12}[N+i] == w_{high} \; AND \; L_{c13}[N+i] == w_{low}) \\
&\Theta[i] = (L_{c11}[i] + L_{c12}[i])/2 \\
&\textbf{if } (L_{c11}[N+i] == w_{high} \; AND \; L_{c12}[N+i] == w_{low} \; AND \; L_{c13}[N+i] == w_{low}) \\
&\Theta[i] = L_{c11} \\
&\textbf{if } (L_{c11}[N+i] == w_{low} \; AND \; L_{c12}[N+i] == w_{high} \; AND \; L_{c13}[N+i] == w_{low}) \\
&\Theta[i] = L_{c12} \\
&\textbf{if } (L_{c11}[N+i] == w_{low} \; AND \; L_{c12}[N+i] == w_{low} \; AND \; L_{c13}[N+i] == w_{high}) \\
&\Theta\,[i] = L_{c13} \\
&\textbf{if } (L_{c11}[N+i] == w_{low} \; AND \; L_{c12}[N+i] == w_{low} \; AND \; L_{c13}[N+i] == w_{low})
\end{aligned}
\tag{1.15}
$$

$\Theta[i] = (L_{c11}[i] + L_{c12}[i] + L_{c13}[i])/3$

if $((L_{c11}[N+i] == w_{med}$ OR $L_{c11}[N+i] == w_{high})$ AND $(L_{c12}[N+i] == w_{med}$ OR $L_{c12}[N+i] == w_{high})$ AND $(L_{c13}[N+i] == w_{med}$ OR $L_{c13}[N+i] == w_{high}))$

$\Theta[i] = (L_{c11}[i] + L_{c12}[i] + L_{c13}[i])/3$

if $(L_{c11}[N+i] == w_{low}$ AND $L_{c12}[N+i] == w_{low}$ AND $L_{c13}[N+i] == w_{med})$

$\Theta[i] = L_{c13}[i]$

if $(L_{c11}[N+i] == w_{low}$ AND $L_{c12}[N+i] == w_{med}$ AND $L_{c13}[N+i] == w_{low})$

$\Theta[i] = L_{c12}[i]$

if $(L_{c11}[N+i] == w_{low}$ AND $L_{c12}[N+i] == w_{med}$ AND $L_{c13}[N+i] == w_{med})$

$\Theta[i] = (L_{c12}[i] + L_{c13}[i])/2$

if $(L_{c11}[N+i] == w_{med}$ AND $L_{c12}[N+i] == w_{low}$ AND $L_{c13}[N+i] == w_{low})$

$\Theta[i] = L_{c11}[i]$

if $(L_{c11}[N+i] == w_{med}$ AND $L_{c12}[N+i] == w_{low}$ AND $L_{c13}[N+i] == w_{med})$

$\Theta[i] = (L_{c11}[i] + L_{c13}[i])/2$

if $(L_{c11}[N+i] == w_{med}$ AND $L_{c12}[N+i] == w_{med}$ AND $L_{c13}[N+i] == w_{low})$

$\Theta[i] = (L_{c11}[i] + L_{c12}[i])/2$

if $(L_{c11}[N+i] == w_{med}$ *AND* $(L_{c12}[N+i] == w_{med}$ *OR* $L_{c12}[N+i] == w_{high})$ *AND* $(L_{c13}[N+i] == w_{med}$ *OR* $L_{c13}[N+i] == w_{high}))$

$\Theta[i] = (L_{c12}[i] + L_{c13}[i])/2$

if $((L_{c11}[N+i] == w_{high}$ *OR* $L_{c11}[N+i] == w_{med})$ *AND* $L_{c12}[N+i] == w_{low}$ *AND* $(L_{c13}[N+i] == w_{high}$ *OR* $L_{c13}[N+i] == w_{med}))$

$\Theta[i] = (L_{c11}[i] + L_{c13}[i])/2$

if $((L_{c11}[N+i] == w_{high}$ *OR* $L_{c11}[N+i] == w_{med})$ *AND* $(L_{c12}[N+i] == w_{med}$ *OR* $L_{c12}[N+i] == w_{high})$ *AND* $L_{c13}[N+i] == w_{low})$

$\Theta[i] = (L_{c11}[i] + L_{c12}[i])/2$

Finally, the values obtained are assigned to the adapting vector corresponding to the A_{c1} cluster vector under observation. Thus, $\forall$ i = 1, 2,…, N:

$$A_{c1}[i] = \Theta[i] \ . \qquad (1.16)$$

The rules-based decisions, as depicted in expressions (1.15) and (1.16), are repeated similarly for all the existing clusters under observation, in this case A_{c2} and A_{c3}; by using the same logic and the existing knowledge within L_{c21}, L_{c22}, L_{c23} and L_{c31}, L_{c32}, L_{c33} cluster occurrences. Furthermore, the adapted values based on the A_{c1}, A_{c2} and A_{c3} are used to replace the previous values (which are added, if not available in the existing knowledge base). Thus, new clusters of data are created, reflecting the adaptability of a system under observation, based on the occurrences. As an example, if A_{c1} to A_{c7} represent daily clusters under observation (i.e., Monday to Sunday), L_{c11}, L_{c12}, L_{c13} are three occurrences during Monday (i.e., cluster vector A_{c1}), L_{c21}, L_{c22}, L_{c23} are three occurrences during Tuesday (i.e., cluster vector A_{c2}), etc.

In order to verify the ALS model, two different scenarios involving pattern changes of heat set point of PCT and also leave time, are considered. Table 3.1 below shows different heat/cool set points (SP) and time of the day default schedules, set by an occupant. Hence,

the scenarios under consideration are the pattern of 'Leave' time and 'Heat' SP pattern changes initiated by the occupant.

Table 3.1 – Daily PCT schedule of temperature SPs

Description	Time of Day (hr)	Heat Set Point (°C)	Cool Set Point (°C)
Sleep	0:00 AM - 6:00 AM	17	19
Wake	6:00 AM - 8:00 AM	18	20
Leave	8:00 AM - 18:00 PM	17	19
Home	18:00 PM - 12:00 AM	20	22

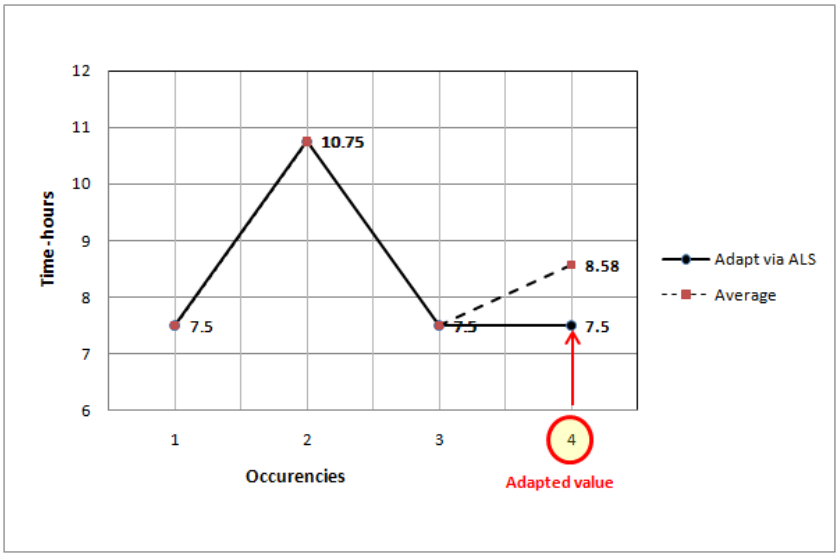


Figure 3.3 – Leave time patter changes adapted via ALS vs. averaging

The graph in Figure 3.3 depicts the adapted 'Leave' time after three consecutive occurrences of occupant's pattern changes (i.e. occupant leaving the house at different times compared to the pre-programmed PCT daily schedule). The tolerances used in this example for the ALS model are depicted in Table 3.2, which are considered while executing the rules-based decisions in a response to the occupant's pattern changes. Fig. 3.3 depicts how the ALS model after few occupant's pattern changes of 'Leave' time (three occurrences in our case), it adapts the leave time value of 7.5 hr (7:30 AM) instead of 8 hr (8:00 AM). In addition, Fig. 3.3 also shows the example of an averaged value (i.e. averaging of the occurrences instead of applying ALS model), which corresponds to 8.58 hr (8:34:48 AM).

Table 3.2 – Tolerances and weights for 'Leave' time and heat set points

Tolerances	Leave time tol. (hr)	Heat SP tol. (°C)	Associated weight
ϵ_{low}	1	1	W_{high}
ϵ_{med}	1.5	3	W_{med}
ϵ_{high}	3	5	W_{low}

Taking into account that occupant's first leave time was at 7.5 hr, second time at 10.5 hr and third time at 7.5 hr, the ALS model adapted value is 7.5 hr, which is closer to the occupant's preferred value. Thus, ALS, whilst adapting to the occupant's preferred values, it also takes into account the energy conservation aspect.

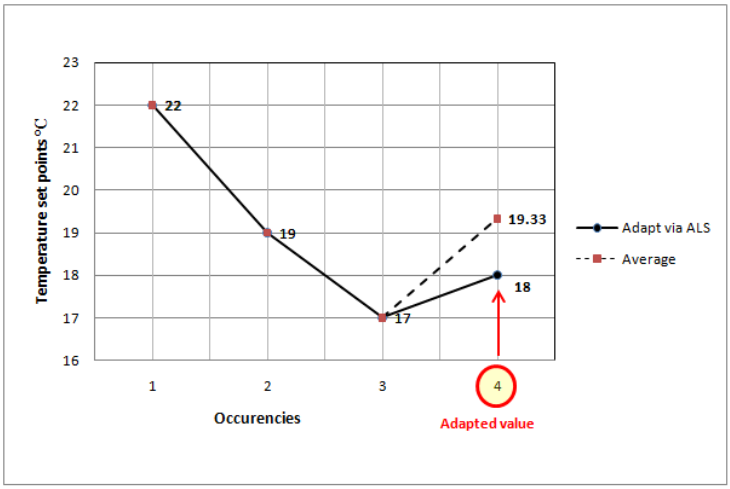


Figure 3.4 – Adapted set points via ALS vs. averaging

In Figure 3.4 are depicted the results of adapted value via ALS model versus average, for three different daily cluster occurrences of the heat SP pattern changes. The first week's 'Wake' heat SP (refer to Table 3.1) was changed from 18 °C to 22 °C, on the second week it was changed to 19 °C, and on the third week it was changed again to 17 °C. If the average of three occurrences was taken, the adapted value would have been 19.33 °C, indicating a shift of 1.33 °C from a typical schedule initially chosen by the occupant. Thus, ALS adapted value of 18 °C, is closer to the occupant's scheduled heat SP, and it also signifies better energy conservation. The ALS Model proposed and discussed above was applied in a practical example, hence a new algorithm: *"Observe, Learn and Adapt (OLA) - A new algorithm for Smart Homes using Wireless Sensors and Artificial Intelligence"* was developed. The OLA algorithm and the tools developed (House Simulator – expert system shell) in order to prove the ALS model, are described in Chapter 5 and 6.

3.3 Synopsis of the Solution

3.3.1 High Level Architecture

The envisioned concept of an *"Adaptive Systemic Solution"* – consists of an adaptive environmental control system, utilizing WSN and AI, which can provide energy management in a Smart Home and/or Building; The WSN consisting of numerous 'intelligent agents' (i.e., smart sensor/actuator nodes) and central controller unit that utilizes rule-based expert system concepts in conjunction with the learning and adaptability, related to the user preferences, rates of heating/cooling of different zones, and added *'system awareness aspect'* for energy savings. The AI to the environmental control problem enables the main controller unit to adapt and learn from the system dynamics, using a rule-based expert system strategy and adaptive learning, which are the building blocks of the main controller unit, as depicted in Figure 3.5.

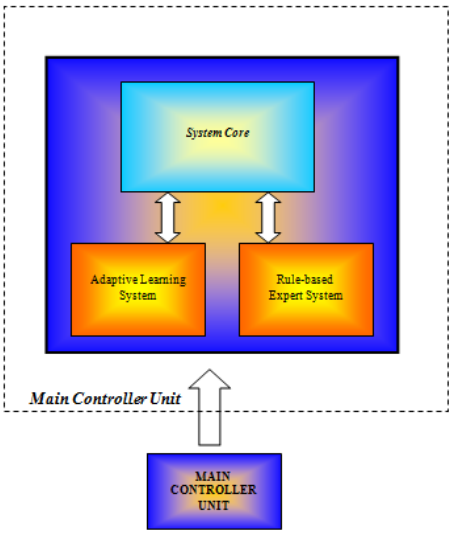


Figure 3.5 - Main controller unit for energy management in intelligent buildings

The main controller unit consists of a "System Core", which includes the ALS and rule-based expert system concepts. Since the rule-based expert system depends on prior knowledge and the amount of time required matching the rule with the memory, using only one AI technique in this case would not render a flexible solution. Therefore, the goal is to blend and utilize AI models based on unsupervised learning strategies, such as the proposed ALS model implementation (via OLA) as described in previous Section 3.2.1 "Adaptive

Learning System (ALS) Model." The capabilities of a system to monitor and control, maintain, and upgrade are also vital components of the overall system and should be taken into account. One of the main features of the primary main control system must be its robustness, ability to adapt and perform autonomous functions, and maintain its operational state in case of faults (system recovery). Also, capabilities to remotely upgrade sensor nodes' firmware over-the-air assist the overall system performance and capabilities.

The main controller unit, in a way should reflect a 'magnified' and more powerful version of a PCT with wireless interface – two way communication and network capability, interface to a multitude of sensors/actuators, offering a variety of options for controlling thermal comfort and/or other appliances. Today's PCTs are able to communicate with the home appliances, electricity generating utility meters - helping in the peak load control (demand response initiative), offering efficient use of energy resources, thermal comfort and energy savings. The integration of PCT devices and multiple wireless sensor networks into Home Automation Systems, contributes in the implementation of "Smart Homes and Buildings". The main controller unit has the ability to access and retrieve information constantly from the sensor nodes placed in many essential parts of the building, managing more closely the HVAC and lighting control systems within that building. Hence, this reduces the unnecessary waste of energy resources when not needed.

The wireless sensors on the other hand, offer the freedom to place sensors in any possible part of the building without the need of wires, while providing abundant reading/monitoring information for the optimal energy management within a building. The capabilities of sensor nodes to sense different variables of interest, such as temperature, humidity, pressure, airflow, occupancy, sunlight and other, could greatly reduce the limitations of the existing energy management systems. Thus, enabling a better management of energy, by using the sensors' data to analyze and control efficiently HVAC system(s) for optimal adjustment of heated and/or cooled air flow in multi-zones via actuators (controlling air valves, fans, etc.). In addition, utilizing indoor and outdoor sensors/actuators to sense the light intensity, open and close windows and/or blinds, in order to optimize the yield with respect to energy conservation and comfort, can apply in this case, as well. By turning lights on/off only when needed (i.e., lighting control system) in specific parts of the buildings, sensors/actuators further contribute in conserving energy. The present and near future

intelligent building designs, should be equipped and/or consider the means to exploit solar energy (i.e., via solar panels), making use of renewable energy resources, as a secondary source of power.

The intelligent building system should be able to take actions based on its *'sensor inputs'* (i.e., sensor/actuator node data) where the overall aggregated and/or compiled information is used to achieve the optimal results associated with the energy efficiency and comfort.

The sensor nodes are capable to communicate among each other and the main controller unit. Whilst, the main controller unit also communicates with the AMI infrastructure (directly or indirectly via central controller unit), in order to monitor the TOU prices/rates and implement the DR Electricity Management, in order to save energy and manage the peak load demands.

In addition to the two-way communication capabilities of the main controller unit and sensor nodes, the ability (via gateways) to access the internet - opens many opportunities to explore the power of WSN and control units within a system (equipped with the state-of-the-art intelligent software) to remotely access, analyze, act and forecast best suitable actions, leading to energy conservation and comfort. Thus, as a result, the ability to communicate with the AMI devices (i.e., Smart Meters) enables thriving implementation of DR Energy Management initiatives, help EGU to reduce the peak load demand, improve the control of energy supply and demand, and reduce the overall energy costs.

The WSN contribute in actuation, sensing and dissemination of the real-time information to the main control system. In a way, WSN represents the "awareness of a building", and the AmI is a result of it. The strategy for creation of such a system, should encompass not only the system automation, AI and AmI, but furthermore must take advantage of a low cost and compact size, yet powerful wireless sensors. A well-thought composition of such a *"systemic solution"* could lead in fact, to a brilliant synthesis of WSN technology, AI techniques and AmI.

3.3.2 Wireless Sensor Networks for Energy Management in Smart Homes and Buildings: 'An Adaptive Intelligent System'

The methodology for creation of proficient *Intelligent Buildings* is by striving to make them more adaptable, autonomous and aware of our environment, flexible and intelligent enough to sense, actuate, compute and evolve into '*An Adaptive Intelligent System'* which is adaptable and re-configurable; can respond and adapt to new changes and requirements by exploiting the wireless sensor/actuator network capabilities, and the overall system intelligence. Thus, providing the means for efficient energy management and helping to accomplish many other essential operations of *Intelligent Buildings,* such as tapping into the renewable energy resources, in addition to the automation and control.

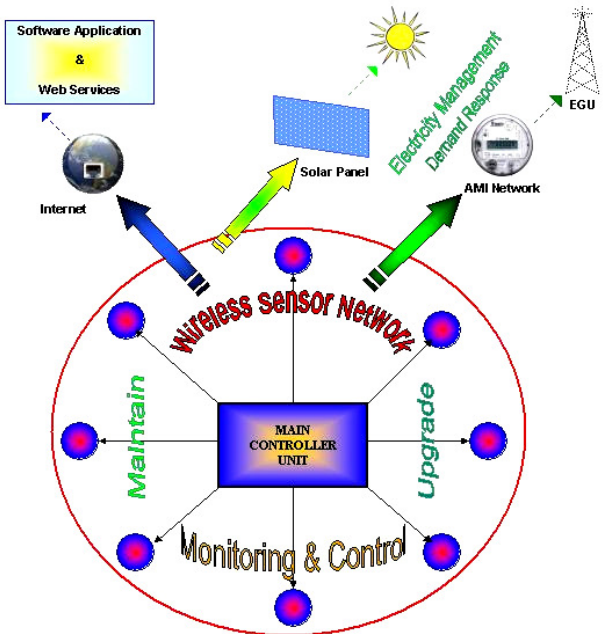


Figure 3.6 - WSN for energy management in smart homes and buildings

The Figure 3.6 depicts a conceptual design of the Energy Management in Intelligent Buildings, by utilizing WSN, AmI, Advanced Control Systems and AI.

The Ambient Intelligence (distributed electronic intelligence) is a modern technology embedded into our natural surroundings with sensing, adaptability and/or reaction to the presence of people and objects. Thus, WSN mutually with the AI and AmI capture the

essential ingredients that could be incorporated into *Intelligent Buildings* in order to accommodate many needs, such as the energy management, security, and/or other needs.

3.3.3 System Integration

Considering that different parts of the building might experience different load profiles and occupancy preferences, having only one centralized control unit for HVAC system for energy management would not suffice. A reasonable approach to the problem is system integration of several sub-systems with dedicated functionality (i.e., based on the size of building and its needs) contributing towards the achievement of a single objective - efficient energy management.

Figure 3.7 depicts a conceptual system design where the system integration and interaction among several sub-systems is considered. The shared effort of the envisioned system to augment its performance is by dissemination of data and control power to the peripheral parts of the system, as required. The idea of system interaction, combined with the power of wireless sensor nodes, strengthens the system ability to react in different scenarios i.e., when different zones within a building require different accommodations (heating or cooling). The overhead in wireless sensor nodes and main control system is removed by the peripheral controller unit taking the role of a '*clusterhead'* or sink within a neighborhood of sensor nodes responsible for specific coverage areas (i.e. zones within a building), to process, aggregate and exchange gathered data with the main control system, in order to achieve the optimal energy management schemes for the entire building, at any given time.

The peripheral controller unit is equipped with high-energy resources (plug-in device) and transmission power, whereas the sensor nodes are equipped with limited transmission range and energy resources (batteries). Therefore, a benefit of peripheral control systems is also in removing the burden from limited energy resources of the sensor nodes (shorter transmission range required i.e. less power consumption) far away from the main controller unit. Thus, reducing the need to replace and/or charge sensors' batteries frequently, leading to less interruption of data feed from the sensors and reducing the system maintenance cost). Thus, the real-time response of the system is improved by faster computations, less error

prone communications, prolonged lifetime of sensor node energy resources, and finally better overall system efficiency.

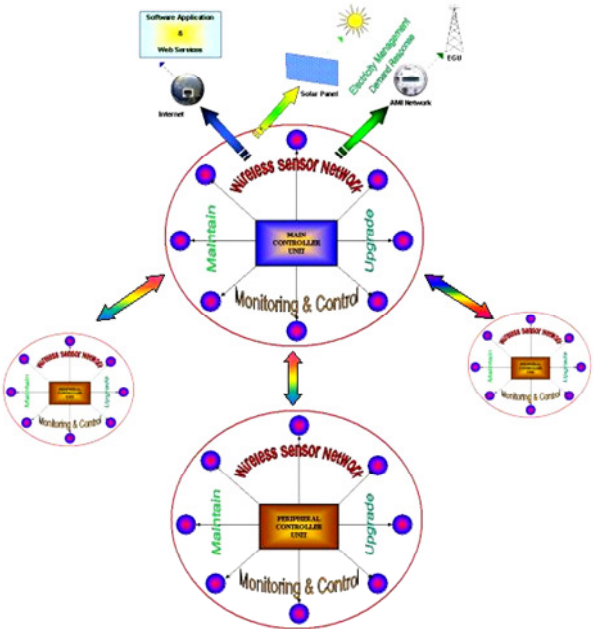


Figure 3.7 - System integration for energy management in smart homes and buildings

Considering all the factors that influence intelligent building system operation, it must be said that the system can accommodate different building requirements with respect to energy savings and management of electricity within it. As described earlier, the role of sensor nodes in this matter is of vital importance. However, based on the size of a building, its requirements, thermal characteristics and geographic position, for better-controlled environment and energy management, system integration encompassing WSN, main controller and peripheral units, leads to a more advantageous intelligent building system design. The emergence of WSN has lead to the advancement in Ambient Intelligence, and has opened an abundant source of opportunities for application of powerful controllers, and AI concepts. The main objective of this section is to convey the key elements of an '*Adaptive Systemic Solution,'* a glimpse of a design paradigm to be considered in energy management system implementations in intelligent buildings, in general. An adaptive system should not only monitor and control, but adapt and optimize the comfort with respect to energy consumption by observing the surrounding and external elements of design, utilizing the power of WSN, learning occupancy patterns and preferences; enabling efficient comfort zone adjustment of the entire building.

"In order to achieve paramount results in our efforts to design an 'Adaptive Systemic Solution,' we must cross the boundaries of a single discipline and consider the problem from a 'Systems perspective.' Where the interaction of several subsystems, each with different attributes and specific qualities are considered - complementing each other as a whole, thus applying and using the gathered knowledge of inter-disciplines, towards the creation of complex intelligent systems."

3.4 Development Tools

To facilitate the proof of concept, in addition to PC, the following software / hardware development tools were used:

- Microsoft Visual C# for the implementation of the House/Thermostat Simulator system and proof of concept for the proposed adaptive learning technique (its implementation) via OLA. Implementation of the RL and Agent-based technique for finding optimal solution in a predefined interval
- CD++ toolkit for Discrete-Event modeling and simulation of wireless sensors and initial Thermostat Simulator System using DEVS and Cell-DEVS
- Embedded development tools (hardware and software)
    - CCS C Compiler for 24 bit PICmicro®MCUs
    - In-Circuit Debugger (ICD-U40)
    - PICmicro®MCU 24FJ128GA006 development board
        - One off board Temperature sensor (DS1631)
        - One on board Digital Potentiometer / Two on board Push Button(s)
        - Three onboard LED(s) / One off board 2x7 segment LED Display
        - Two off board LEDs and one off board Push Button

CHAPTER 4

FINDING GLOBAL MAXIMUM IN A PREDEFINED INTERVAL

In this chapter, a new algorithm for use in embedded controllers with limited memory, processing power and/or energy resources for finding the global maximum of a function in a predefined interval, by using the reward/punish concepts from reinforcement learning and agent based techniques is presented. *"Reinforcement Learning and Agent-based Search"* application was implemented in C# to observe the algorithm at work and demonstrate its main features. The performance results for several different functions are presented in order to demonstrate the efficiency of the algorithm at work. Thus, its usefulness in embedded systems with limited memory and/or processing power, such as the wireless sensor and/or actuator nodes is discussed.

4.1 Introduction

In many real world applications use of only one technique to solve a particular problem will not suffice to bring forward the best possible solution. Hence, the combination of different techniques has led to the emergence of more sophisticated intelligent hybrid systemic solutions, which combine at least two different artificial intelligent (AI) techniques and/or intelligent technologies. The change of focus from weak to domain specific methods within AI, lead to exploration and discovery of biologically inspired techniques such as ANN, GA, simulated annealing, PSO, RL, intelligent agents, etc. The RL and agent-based computing, as a part of AI techniques, has shown to be capable of solving complex real world problems in vast engineering and computer science applications [POS08][SON98] [TAN09]. The GA and PSO are also known to be useful in solving optimization problems [IZQ08]. In addition, the use of multi-agent based approaches is apparent also in wireless sensor networks, where sensor and/or actuator nodes act as intelligent agents cooperating among themselves to achieve a particular objective. The advantages of efficient problem solvers are numerous, however, in our case, of particular interest are the problems which are closely related to small-scale embedded systems, or better say sensor/actuator nodes with

limited computational power, memory and energy resources, implying that fast and efficient processing algorithms, should be considered as an essential part of the overall system design.

The role of DR and TOU rate incentives applied by utilities (EGUs), have great impact in peak load curtailment, and are used to control the peak load demands to be able to cope with ever-increasing energy demands and available resources i.e. capacity which is limited due to the number of generators available. On the other hand, DR and TOU incentives help the consumers to save on high electricity prices.

However, there are many parameters which play a role in achieving a peak load curtailment during the cold winter and/or hot summer days. The control of power usage in homes and buildings is not related just to HVAC systems and appliances; it includes the load control switches for water heaters, pool pumps, hybrid electric vehicles, lighting etc. Thus, the pattern of electricity usage by consumers in general, is not constant. Different parts of a city and/or province might have different power usage patterns, where the control of peak load demand, poses a challenge to predict and control. Different zones within a city might have to Opt-in (engage) and Opt-out (disengage) to DR commands and TOU rates, at different times of a day, based on the demand for electricity.

The dynamic nature of a problem at hand implies that utilities must forecast many possible scenarios, based on the weather data and/or previous knowledge-base history in order to respond quickly, better optimize and control the power usage during peak load periods. The utilities communicate (via Smart Meters) with “Smart Thermostats”, in order to manage the usage of power during peak load events, accordingly. Taking into account the randomness of possible power usage patterns, hence peak load demands, utilities can send to each zone of a city several possible predefined functions, reflecting forecasted daily power usage; hence, providing information to the consumer about the DR and high TOU rate prices about the power usage during the peak load periods. The “Smart Thermostats” share this information with the existing WSN in a building and/or home, to better manage the power (i.e. pre-heating and/or pre-cooling of zones), based on the patterns (and prices) extracted from the functions delivered by the utility.

Thus, the main benefits of the proposed algorithm described herein, are to ‘bridge a gap’ between the EGUs (i.e., utilities) and “Smart Thermostats” into “Smart Grid” initiatives. While, from the “Smart Environment” perspective, benefits are that the “Smart Thermostats”

and/or WSNs (sensor/actuator nodes) are not required to receive and store hundreds or thousands of daily data in memory for many possible daily energy usage patterns. But, can easily extract the information from functions (i.e. delivered by the utility) describing those daily patterns. The "Smart Thermostat" and/or sensor nodes can process the information from functions, find the optimal points (i.e. global and local maxima), make decisions and act accordingly. Thus, helping utilities to better manage and control the power (i.e. load curtailment) during high peak load periods, and furthermore provide energy savings for consumers.

The algorithm is intended for use in sensor/actuator nodes i.e., microcontrollers with limited memory, processing power and/or energy resources. Thus, the objective is to find the global maximum value of a function in a predefined interval with minimal number of function evaluations and iterations, given a function $y = f(x)$.

4.2 Description of the Algorithm

The main idea adopted for the algorithm is based on the reinforcement learning (feedback evaluation) and agent based techniques. The feedback evaluation of an agent interacting with the environment provides a means to learn optimal agent's action policies, which are evaluated via reward and punishment. This feedback mechanism aids in the 'awareness' of the agent about success and failure rates of its actions and its performance.

Utilities can provide many possible peak load profiles for any particular day, from which "Smart Thermostats" and/or sensor nodes can extract the possible peak load profile of critical periods and conform to the DR and TOU rate incentives, accordingly. Based on the granularity needed to properly evaluate a given function, a few hundred or even thousands of function evaluations could be needed to determine the peak load profiles of a day (i.e. to find global and local optima). In this case, sequential search techniques are impractical. On the other hand, the uncertainty in finding the optimal points, would pose a problem if the GA approach was used. Since, it would require additional verification i.e. comparison of the results of same function from different runs, under different mutation rates to ensure that the actual optimal points are found. Thus, demanding more processing power by the sensor/actuator nodes. Furthermore, based on the dynamic nature of power distribution and usage, potential functions reflecting the possible peak load profiles of a day (i.e. provided by

utility), can be numerous, and different each time. Therefore, it cannot be assumed that there is only one-time daily processing of a given function.

The use of RL concepts, such as the action selection mechanism, where the action evaluation (i.e., reward or punishment) comprise the basic strategy considered for the problem at hand. However, the algorithm described therein, builds on the synergy of concepts from the RL and agent-based approaches. It introduces four guidance methods: *Reward*, *Discover*, *Alert*, and *Optimal,* and '*Create Agents'* concept (described in Section 4.3). The 'Reinforcement Learning and Agent-based Search' algorithm can be used in embedded systems with limited memory, power and processing resources, such as wireless sensor nodes. On the other hand, other methods, such as the sequential search techniques would demand more resources and processing power in order to find the global maximum of a function. Similarly, use of GA techniques in this case would prove impractical, due to the lack of generalization i.e., need for specific implementation required for each class of problems considered at hand.

The 'Reinforcement and Agent-based Search' algorithm could be useful for scenarios such as the 'intelligent agents' which are required to perform autonomous operations, involving decision making, prediction, learning capabilities and/or adaptability. Hence, the main sink node of the system (core controller unit) could feed the information by only a few characters, representing the function to be evaluated. Furthermore, hundreds or thousands of *P(x, y)* values could be expensive to transmit and require a large amount of memory on the sensor/actuator side, in comparison to transmitting several characters needed to represent different functions. Additionally, sensor/actuator node could extract the function under consideration (by parsing the string) and executing the algorithm. Therefore, find the optimal points for further processing and/or autonomous decision making without a need to receive and store large amounts of information. The reinforcement component of the algorithm is implemented as a simple rule consisting of the following enumeration (named '*Judge*') and corresponding 'guidance methods': *Reward*, *Discover*, *Alert*, and *Optimal*.

Figure 4.1 depicts the main 'guidance methods' used in the algorithm. Initially, the 'guidance methods' - *Reward*, *Discover*, *Alert* and *Optimal* are described. Additionally, the synergy of reinforcement learning and agent-based search techniques is shown by the use of new concept 'Create Agents' in addition to the 'guidance methods'.

4.2.1 Reward Method

Reward is one of the 'guidance methods' which moves the point forward sequentially while the function is increasing (refer to Figure 4.1, point P_0 moving towards new position i.e. point P_1). The *Reward* method, only the first time has a step size of one, and increments by two and more (if necessary for accelerated search of the space). The ability not to miss the maximum points, even though the step size is larger is by collaboration with the other methods (i.e. *Alert*, *Discover* and *Optimal*). If during the search no optimal point is found and the limit of the subinterval is reached a flag is set to indicate that the boundary is reached.

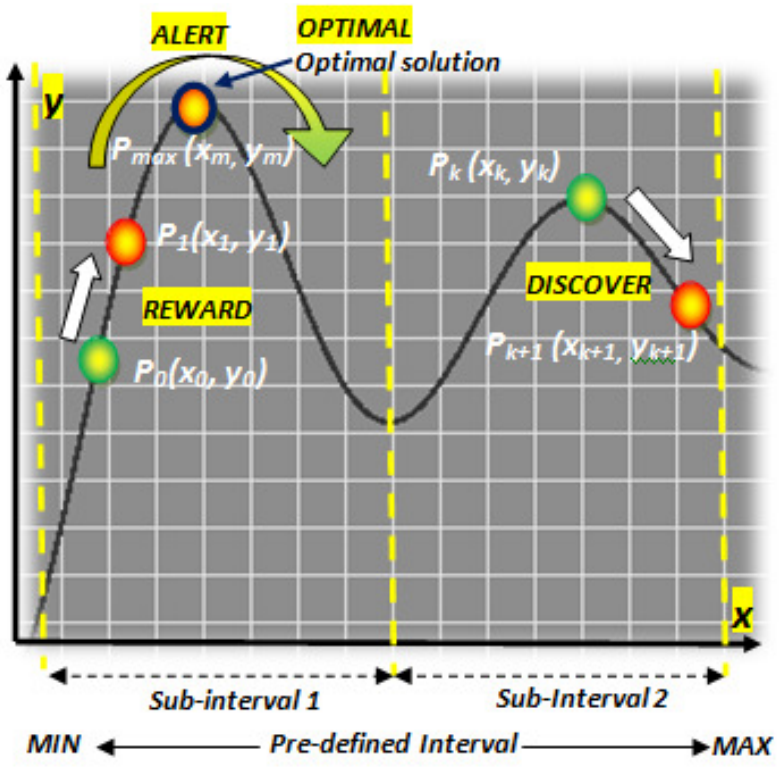


Figure 4.1 - Graphical view of the main concepts

4.2.2 Discover Method

Discover method is executed when the function is decreasing, otherwise it is analogous to the *Reward* method (refer to Figure 4.1, point P_k moving towards point P_{k+1})

4.2.3 Alert Method

Alert method is executed if the global or local maximum is close or is found (i.e., function goes from increasing to decreasing within the subinterval, as depicted in Figure 4.1). If the previous step (prior to execution of the *Alert* method) was *Reward*, step size is reduced accordingly to what the increment step size was performed by *Reward* method, and the consequent movements of the point are made slowly, until the optimal point is reached. For each step, the *P(x, y)* point is evaluated if it is within the ±*Tolerance* limits set at the initial stage, and if true, *Optimal* method is executed and the point P_{max} (x_m, y_m) is recorded (refer to Figure 4.1). Otherwise, if the optimum point is not reached on the first trial, *Alert* moves slowly until the point is reached.

4.2.4 Optimal Method

As shown by former *Alert* method, *Optimal* method is executed during tracking for the global maximum point of the function. The *Optimal*, checks if the point exists and if not it records the point as a new point found. The optimal point indicates to the search method that the global maximum point is found, hence the subinterval completes (this gives flexibility to define subintervals and/or different implementation techniques of the algorithm). One could divide the function interval into several subintervals, approximately sufficient not to miss any maximum (local or global). Otherwise, one could record the coordinates of global maximum point, and adjust the minimum value of the next subinterval to the optimal point value plus given *Tolerance*, and the maximum value to the previous maximum or a new maximum; taking into account that the new maximum of the subinterval is less than the maximum of the entire predefined interval (refer to Figure 4.1, subinterval 1, subinterval 2 and the predefined interval). Furthermore, in Section 4.3 it will be described how the synergy between RL i.e., 'guidance methods' and agent-based techniques, provides a more effective solution to the above described scenario.

4.2.5 Main Steps of the Algorithm

4.2.5.1 Problem Definition

Given function $y = f(x)$, where $x \epsilon$ [a, b] find maximum value of a function in a predefined interval [a, b] with minimal number of function evaluations and iterations.

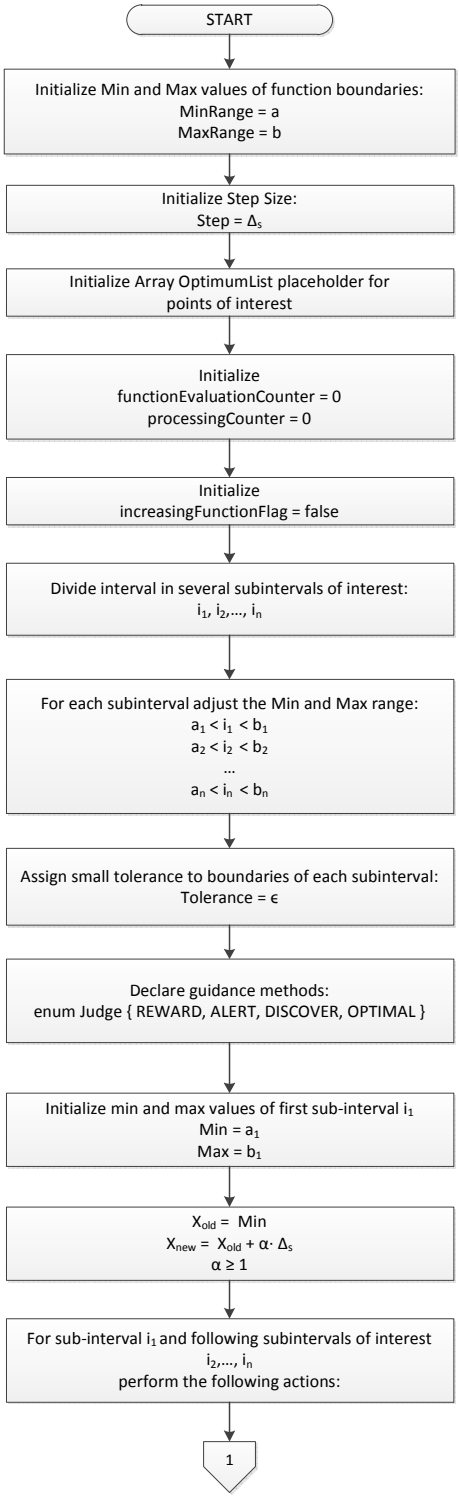
4.2.5.2 Algorithm

The main steps of the algorithm are depicted in Figure 4.2. The interval under observation, is divided into several subintervals of interest, and accordingly, for each subinterval the minimum and maximum range is adjusted (total should be equal to the predefined interval). In addition to the 'guidance methods' described earlier, few methods, such as *SearchProcess* and *CheckBoundaryPoints* are discussed below.

The *'SearchProcess'* method executes while the point of interest is not found or the maximum range of the subinterval is not reached. During each execution, function is evaluated and checked against the old and new x values, and *increasingFunction* flag is assigned accordingly. If the function under consideration is increasing i.e. function value is greater for the input value *Xnew* versus the value *Xold* then *increasingFunction* is true, otherwise it is false. Additionally, for each specified subinterval method *SearchProcess* is executed, while keeping track of the reference points and values for the old and new steps taken.

The *CheckBoundaryPoints* method checks each subinterval boundary points using the references to *Xold* and *Xnew* values by evaluating the function under consideration. Thus, marks the *increasingFunction* as true if the function is increasing and false otherwise. If the boundary of the search space is reached (maximum predefined interval value) method terminates, otherwise consequent subinterval is executed following the same principles. Furthermore, the method *'Create Agents'* is described in Section 4.3. The step size Δ_s can be chosen based on the granularity of a function under consideration. Moreover, the term α is the tuning factor of the step size Δ_s (i.e. $X_{new} = X_{old} + \alpha \cdot \Delta_s$), which directly affects the search process (i.e. $\alpha > 1$ accelerate the search, and $\alpha < 1$ decelerate). When the search process is in the accelerated mode, and transition from the *Reward* to *Discover* state happens while no optimal point is detected, the *Alert* state becomes active and moves back to the step which

was active prior to the *Discover* state. Thus, it ensures that no optimal point is missed during the search process. Furthermore, while in the *Alert* state, value of α is decreased (i.e. $\alpha < 1$) and is neither modified nor updated until the optimal point is found.



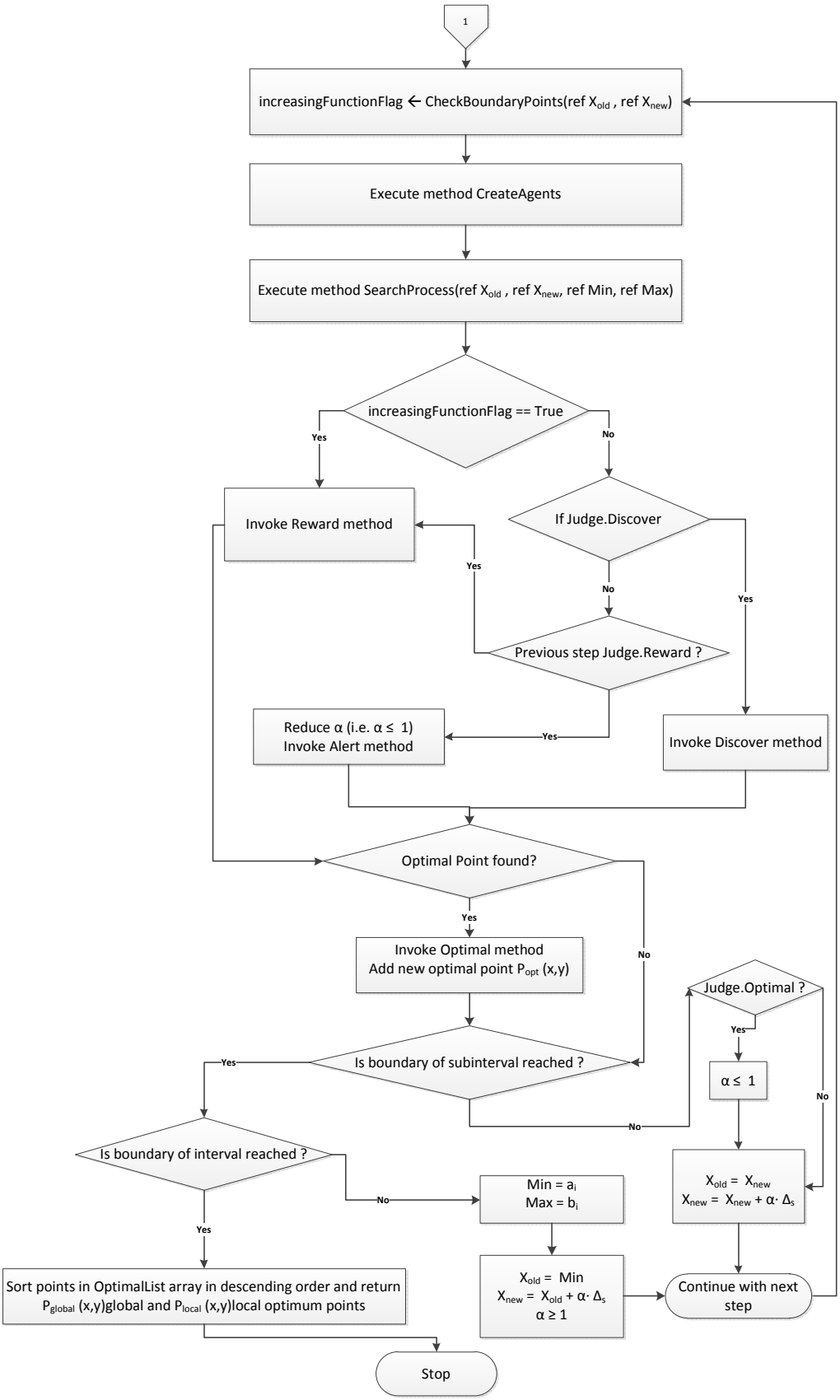


Figure 4.2 – Basic flowchart of the RL and Agent-based search algorithm

4.3 Synergy of Reinforcement Learning and Agent-Based Search Technique

The agent-based technique, in addition to the reinforcement learning i.e. 'guidance methods,' randomly generates a cluster of agents, whereas each point *P(x, y)* is evaluated for its fitness. Taking into account that we are looking for the maximum value (i.e., global maxima) of a function within a specified interval [a, b], point $P_{max}\ (x_m, y_m)$ which gives the highest value is chosen as a starting point of the search process. It will be shown (in Section 5) with tangible simulation results that agent-based technique complements very well the reinforcement learning strategy applied to this problem, and indeed it offers much better results in comparison to the reinforcement 'guidance methods' alone (referred also as "Reinforcement Only"). The addition of a new concept '*Create Agents'* was introduced to the previously discussed algorithm, in order to make it more flexible and efficient in finding the optimal solution (i.e., global and local maxima), for much shorter processing time and with less functional evaluation overhead. The basic principle applied is the following: Based on the function $y = f(x)$ under investigation, agent-based concept involves creation of generation of agents for the given predefined interval [a, b]. The list of agents is sorted in ascending order, and the best agent, i.e. the one that produces maximum value for the function $y = f(x)$ in a sub-interval $[a_1, b_1]$ where $a \leq a_1$ and $b_1 \leq b$, is chosen as the initial starting point of the *SearchProcess* method described priori. Agent size being used can be set to any number; however it was observed that agent size of 10 would suffice for most of the problems with small interval range. Predefined interval can be divided in several sub-interval based on the granularity of the solution space required, and the functions being evaluated. It is not prudent to use unnecessarily large number of subintervals for problems where the solution can be achieved with only a few.

4.4 Reinforcement Learning and Agent-Based Search Application

The application software *"Reinforcement Learning and Agent-based Search"* was implemented in C# in order to evaluate and demonstrate the algorithm proposed, by simulating the algorithm at work (for several functions) under different scenarios, in

particular 'Reinforcement Only', in isolation from the 'Agent-based Search' technique applied, and finally the synergy of the reinforcement learning and agent-based technique.

4.4.1 Graphical User Interface – Main Features

On the right hand side of the application GUI (refer to Figure 5.3), functions being evaluated are displayed, including the points of interest. While on the left hand side of the GUI, function being evaluated, its range and step size used, following which are displayed the output results including the optimum points found (and their coordinates), number of iterations taken during this process, including the number of function evaluations during the search process. Clicking on the '*Reinforcement Only'* button invokes the Reinforcement algorithm whilst, clicking on the '*Reinforcement and Agent'* button, invokes the Reinforcement and Agent-based search algorithm, for the selected function under test, which gets executed. The output results are displayed in the output text window and also drawn in the graph.

4.5 Performance Results

Several functions with different characteristics were tested in order to verify the algorithm, its main features; and also to compare the "Reinforcement Only" versus "Reinforcement Learning and Agents-based Search" techniques. The first function considered is depicted in Figure 4.3. Function*: $y = 2\ ((\cos(\pi x) / x) + 1)$* where $x \in [1, 5]$; Step size = 0.005; Tolerance = 0.05; $P_{max}\ (x, y)$ to be found;

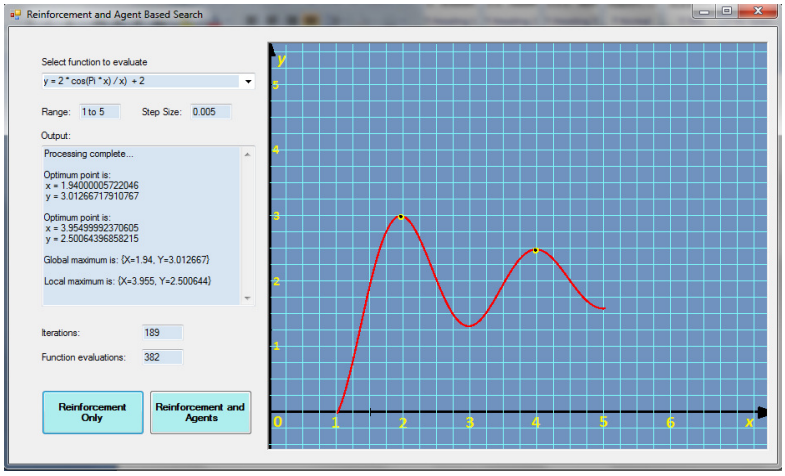


Figure 4.3 - Reinforcement learning and agent-based search application

From the results shown in Table 4.1, it can be observed that by applying the 'Reinforcement Only' technique, the maximum is found after 189 iterations, while the number of function evaluations is 382. However, the results depicted in Tables 4.2 indicate better performance of the "Reinforcement Learning and Agent-based Search" when compared to the "Reinforcement Only" technique. As shown in Tables 4.2, for the "Reinforcement Learning and Agent-based Search" technique, the maximum value of a function was found after 30.8 iterations (calculated from 10 trials) versus 189 for the "Reinforcement Only". Similarly, the number of function evaluations was found to be 109.2 versus 382. The effectiveness of "Reinforcement Learning and Agent-based Search" is likely not to be affected by the step size as much as the "Reinforcement Only" technique. Thus, it indicates a much better performance, and validates the optimality of former against the later technique.

Table 4.1 - Simulation results for 'Reinforcement Only'

Simulation Trial	Iterations	Nr. Of Function evaluations	Global max	Local max
1	189	382	{X=1.94, Y=3.012667}	{X=3.955, Y=2.500644}

Table 4.2 - Simulation results for 'Reinforcement Learning and Agent based Search'

Simulation Trial	Iterations	Nr. Of Function evaluations	Global max	Local max
1	18	71	{X=1.934448, Y=3.01204}	{X=3.954919, Y=2.500636}
2	33	111	{X=1.934448, Y=3.01204}	{X=3.958009, Y=2.500914}
3	14	70	{X=1.939554, Y=3.012628}	{X=3.956368, Y=2.500772}
4	49	155	{X=1.935907, Y=3.012235}	{X=3.956165, Y=2.500754}
5	28	90	{X=1.937366, Y=3.012409}	{X=3.959194, Y=2.501008}
6	20	91	{X=1.942644, Y=3.012856}	{X=3.960644, Y=2.501114}
7	48	154	{X=1.940496, Y=3.012708}	{X=3.953733, Y=2.500517}
8	36	129	{X=1.935496, Y=3.012182}	{X=3.96038, Y=2.501095}
9	51	152	{X=1.94421, Y=3.012935}	{X=3.957746, Y=2.500892}
10	11	69	{X=1.939899, Y=3.012659}	{X=3.960775, Y=2.501123}
Average	30.8	109.2		

In addition, the error of maximum value P_{max} *(1.95, 3.01)* is less than the set Tolerance (0.05). Noticeably, introduction of the new concept *'Create Agents'* within the algorithm offers competitive performance advantages to the algorithm. The scenario shown above verifies the effectiveness of algorithm for finding optimal solution in a predefined interval.

However, further scenarios are considered, in order to show its usefulness in real-world applications, as mentioned earlier, in the Introduction section of this Chapter. Thus, in order to show the algorithm at work for different situations (related to the utility's peak load demand), several possible scenarios are considered and verified, as depicted in Figures 4.4, 4.5 and 4.6. The functions being considered reflect different peak load profiles described by polynomials of 6^{th} degree. The x-axes of the graph represents the time of a day 0 to 24 hours, while y-axes represents the probable utility load profiles of energy usage (in percentage) at different times of a day. During cold winter and/or hot summer days, the demand for electricity can be higher that the supply, due to the number of generators available and/or active at a time. The existing infrastructure is getting older and cannot keep up with the ever-increasing electricity demand (i.e. new generators are not build as the demand for electricity grows). In addition, utilities are more interested in balancing the power profile (i.e. peak load curtailment), rather than activating and/or building new generators just to meet the low occurring extra energy demands (i.e. several occurrences per month), needed possibly just for few hours a day.

Figure 4.4 depicts the first load profile of the utility function y_1 under consideration.

$y_1 = 0.941\,x^6 - 7.3795\,x^5 + 21.2895\,x^4 - 28.0191\,x^3 + 16.540\,x^2 - 2.8859\,x + 0.6294.$

Where $x \in [0, 2.4]$; Step size: 0.005; Tolerance = 0.05; $P_{max}\,(x, y)$ to be found;

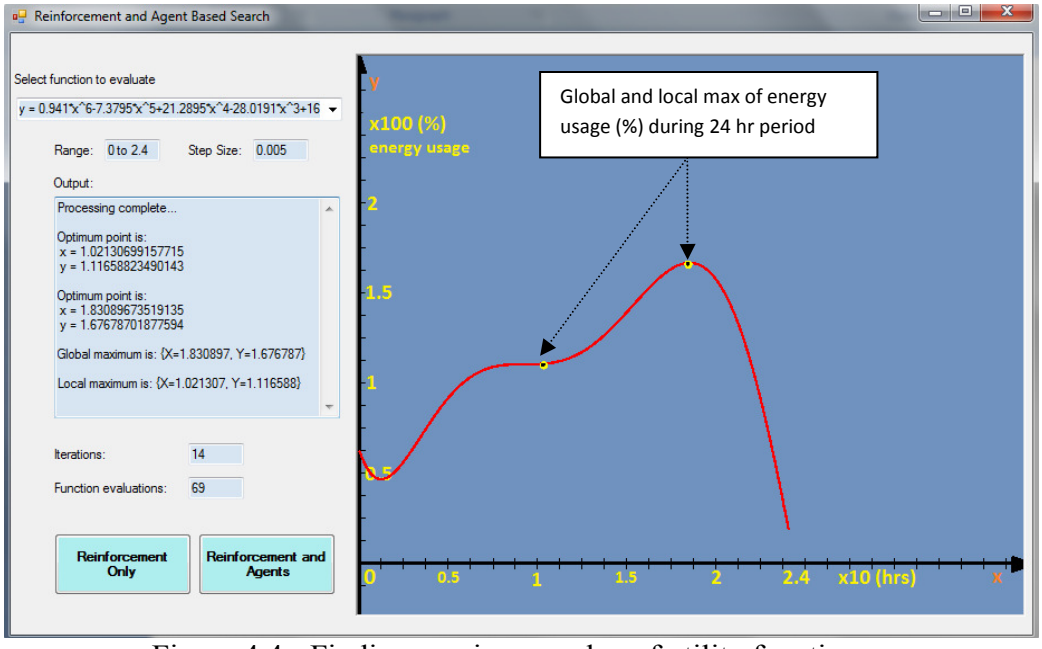


Figure 4.4 - Finding maximum value of utility function y_1

Table 4.3 - Simulation results for 'Reinforcement Only'

Simulation Trial	Iterations	Nr. of Function evaluations	Global max		Local max	
			Time of day (hr)	Energy usage (%)	Time of day (hr)	Energy usage (%)
1	126	256	18.3	167.6694	8.3	111.2027

Table 4.4 - Simulation results for 'Reinforcement Learning and Agent based Search'

Simulation Trial	Iterations	Nr. of Function evaluations	Global max		Local max	
			Time of day (hr)	Energy usage (%)	Time of day (hr)	Energy usage (%)
1	16	73	18.3144	167.6841	9.88252	111.4962
2	15	74	18.3603	167.7214	8.86443	111.4060
3	23	85	18.2849	167.6525	10.01474	111.5465
4	6	58	18.2870	167.6550	10.12052	111.6004
5	14	74	18.3879	167.3370	10.14696	111.6160
6	22	84	18.3308	167.6991	10.06763	111.5718
7	8	54	18.2925	167.6612	9.61808	111.4410
8	6	56	18.3114	167.6811	9.68419	111.4501
9	9	54	18.3682	167.7263	9.72386	111.4569
10	14	69	18.3089	167.6787	10.2130	111.6588
Average	**13.3**	**68.1**				

From the results depicted in Tables 4.3 and 4.4 we can observe the global maximum of the energy usage (%) at specific time of a day (shown in hours). Similarly to the previous case, the results indicate clearly better performance of the 'Reinforcement Learning and Agent-based Search' versus 'Reinforcement Only' for the utility function y_1 being evaluated. The improvement factor is 9.47 for the number of iterations (i.e. 13.3 versus 126 iterations) and 3.75 for the number of function evaluations (i.e. 68.1 versus 256). The maximum value P_{max} with respect to time of a day (hrs) and energy usage (%) is (18.5, 167.74). The error of maximum value found via algorithm P_{max} (18.3, 167.67) is much less than the set Tolerance (i.e. time of a day: 0.05 x 10 = 0.5 hrs, energy usage: 0.05 x 100 = 5 %). For the utility's load profile y_1, based on the results obtained, the estimated global peak (maximum power load demand) is most likely to occur at 18:18 PM ± 30 min. Where the electricity demand exceeds utility's capacity by 67 % (global maximum). During this time period, the energy usage patterns leading to minimum energy usage can be applied (i.e. "Smart Thermostat" tolerance settings leading to maximum conservation), while consumers help utilities to manage the peak load demand by their participation in DR and TOU rate incentives.

Additionally, during the time periods, when energy usage is about 11% above the capacity (i.e. local maximum, refer to the Tables 4.3 and 4.4), medium tolerances can be applied, to help utilities in peak load management, and consumers to achieve energy savings, respectively.

Figure 4.5 depicts the second load profile of the utility function y_2 under consideration.

Function: $y_2 = 0.946\,x^6 - 7.3805\,x^5 + 21.275\,x^4 - 28.067\,x^3 + 16.492\,x^2 - 2.8414\,x + 0.6194$.

Where $x \in [0, 2.4]$; Step size: 0.005; Tolerance = 0.05; $P_{max}\,(x, y)$ to be found;

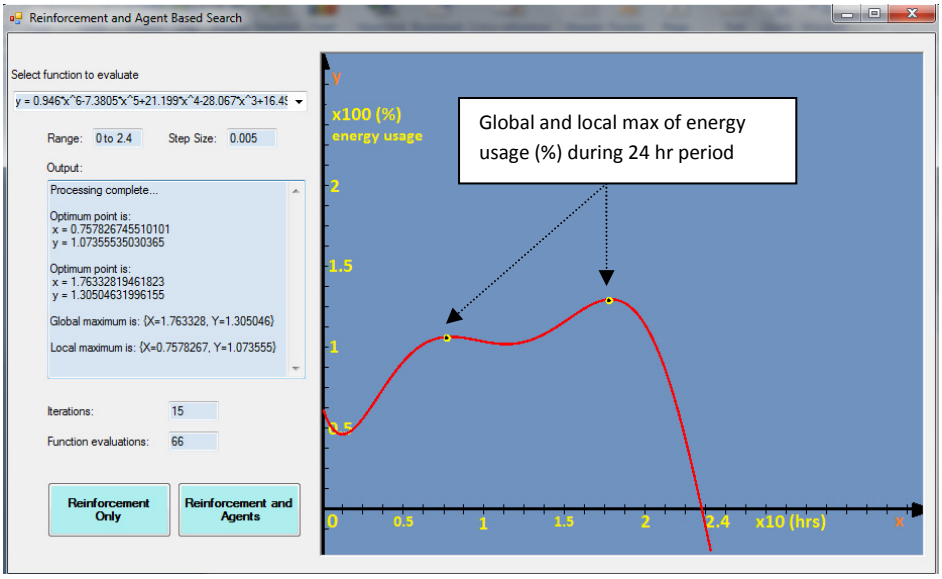


Figure 4.5 - Finding maximum value of utility function y_2

Table 4.5 - Simulation results for 'Reinforcement Only'

Simulation Trial	Iterations	Nr. of Function evaluations	Global max		Local max	
			Time of day (hr)	Energy usage (%)	Time of day (hr)	Energy usage (%)
1	112	228	17.6	130.4727	7.6	107.3728

The results shown in Tables 4.5 and 4.6, reveal the performance consistency of the 'Reinforcement Learning and Agent-based Search' versus 'Reinforcement Only' for the utility function y_2 being evaluated. The improvement factor is 7.46 for the number of iterations (i.e. 15 versus 112 iterations) and 3.02 for the number of function evaluations (i.e. 75.3 versus 228). The maximum value P_{max} with respect to time of a day (hrs) and energy usage (%) is (17.8, 130.57). The error of maximum value found via algorithm P_{max} (17.68, 130.5) is much less than the set Tolerance (i.e. time of a day: 0.05 x 10 = 0.5 hrs, energy usage: 0.05 x 100 = 5 %). For the utility's load profile y_2, based on the results obtained, the

estimated global peak (maximum power load demand) is most likely to occur at 17:48 PM ± 30 min. Where the electricity demand exceeds utility's capacity by 30.578 % (global maximum). Still, in this case maximum and/or medium tolerances leading to energy conservation can be applied by "Smart Thermostats" (around the peak period 17:48 PM), based on the utility's load curtailment response due to the energy load profiles associated i.e. within different zones of a city.

Table 4.6 - Simulation results for 'Reinforcement Learning and Agent based Search'

Simulation Trial	Iterations	Nr. of Function evaluations	Global max		Local max	
			Time of day (hr)	Energy usage (%)	Time of day (hr)	Energy usage (%)
1	25	88	17.6825	130.5416	7.72735	107.4475
2	30	118	17.6554	130.5228	7.65304	107.4094
3	7	56	17.6819	130.5412	7.608359	107.3791
4	17	78	17.6767	130.5379	7.617477	107.3858
5	13	64	17.6010	130.4738	7.767022	107.4616
6	7	56	17.6332	130.5046	7.780243	107.4654
7	14	73	17.6846	130.5429	7.589818	107.3649
8	12	64	17.6360	130.5070	7.614589	107.3837
9	23	90	17.6983	130.5508	7.632827	107.3964
10	15	66	17.6332	130.5046	7.578267	107.3555
Average	**15**	**75.3**				

The local maximum point found for energy usage is about 7% above the existing capacity during the early morning hours (7:42 AM). Thus, utilities may decide to shed the load among many zones of a city, whereas minimum tolerances can be applied by "Smart Thermostats". Otherwise, if the load is shared among few zones of a city, medium tolerances can be applied in order to balance the power usage to less than or equal to 100 %.

Moreover, in Figure 4.6 another load profile expressed by the utility function y_3 is shown for comparison.

Function: $y_3 = 0.7325\,x^6 - 5.6931\,x^5 + 16.597\,x^4 - 22.278\,x^3 + 13.183\,x^2 - 2.1656\,x + 0.5997$.

Where $x \in [0, 2.4]$; Step size: 0.005; Tolerance = 0.05; $P_{max}\,(x, y)$ to be found;

The results depicted in Tables 4.7 and 4.8, for the 'Reinforcement Learning and Agent-based Search' versus 'Reinforcement Only' for the utility function y_3 being evaluated, which indicate a global maximum very close to the local maximum. The algorithm is able to find the optimal point for energy usage, and not get trapped in a local maximum. The

improvement factor is 5.94 for the number of iterations (i.e. 15 versus 112 iterations) and 2.48 for the number of function evaluations (i.e. 18 versus 108). The maximum value P_{max} with respect to time of a day (hrs) and energy usage (%) is (18.6, 106.15).

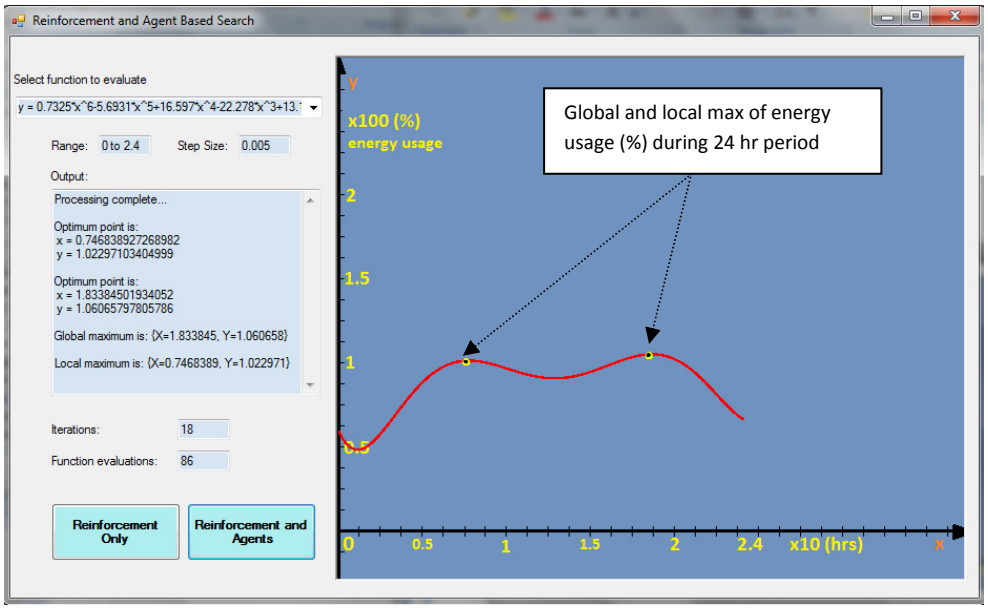


Figure 4.6 - Finding maximum value of utility function y_3

Table 4.7 - Simulation results for 'Reinforcement Only'

Simulation Trial	Iterations	Nr. of Function evaluations	Global max		Local max	
			Time of day (hr)	Energy usage (%)	Time of day (hr)	Energy usage (%)
1	107	220	18.35	106.0736	7.4	102.2437

Table 4.8 - Simulation results for 'Reinforcement Learning and Agent based Search'

Simulation Trial	Iterations	Nr. of Function evaluations	Global max		Local max	
			Time of day (hr)	Energy usage (%)	Time of day (hr)	Energy usage (%)
1	38	113	18.3715	106.0872	7.40805	102.2507
2	23	93	18.2764	106.0180	7.42294	102.2632
3	18	77	18.2898	106.0291	7.49483	102.3140
4	26	96	18.2983	106.0364	7.46504	102.2948
5	17	79	18.3305	106.0603	7.43981	102.2765
6	20	80	18.3497	106.0734	7.39680	102.2409
7	19	87	18.2983	106.0360	7.56869	102.3504
8	12	66	18.3439	106.0695	7.47492	102.3015
9	31	110	18.2952	106.0336	7.45775	102.2897
10	18	86	18.3384	106.0658	7.46838	102.2971
Average	**18**	**88.7**				

The error of maximum value found via algorithm $P_{max}(18.3, 106.07)$ is again within set limits, much less than the set Tolerance (time of a day: 0.05 x 10 = 0.5 hrs, energy usage: 0.05 x 100 = 5 %). In this scenario, since both global and local energy usage peak loads (shown in Figure 4.6) are slightly above the available capacity, utility might decide to send the DR and/or TOU rate incentives, such as to spread the load among all the zones within a city. Thus, having minimal impact on consumers (i.e., requiring minimum adjustment of tolerance settings via "Smart Thermostat"), while achieving the load shedding during the critical high demand times of a day.

4.6 Summary

In this chapter, a new algorithm for finding the optimal solution in a two dimensional space within a predefined interval is presented. The synergy of Reinforcement Learning and Agent-based technique was found to be practical. It was demonstrated with the simulation results as a viable and flexible technique to be used in finding global and local maximum of different functions.

A number of example functions were used to validate its performance with respect to the number of iterations and function evaluations, including possible real-world scenarios, which can be implemented by utilities in order to improve the control of power usage via peak load DR and TOU rate incentives. From the "Smart Grid" perspective, the algorithm 'bridges a gap' and offers a viable solution for further exploitation of "Smart Environments" by utilities, to better manage the peak load demand while offering the consumers options to save on ever-increasing energy costs. Hence, the role of utilities in potential future implementation of such strategies is of vital importance, to further enhance the existing peak load control methods, and/or their efficiency.

During the simulation process, neither the algorithm nor the application code needed to be modified, hence it provides good foundation as a sound method for general use, where no particular implementation is necessary to evaluate and find solutions for different problems. Its usefulness in scenarios such as those of wireless sensor nodes with limited memory and processing power was discussed. The proposed algorithm was implemented, tested and evaluated for finding global maximum of a function in a predefined interval.

CHAPTER 5

SIMULATOR OF A HOUSE HEATING-COOLING SYSTEM

5.1 Introduction

The simulator of a house heating-cooling system was implemented using C# in order to simulate different scenarios of house heating/cooling system efficiency, energy consumption and associated costs under different scenarios. And more importantly for its use as an "expert system shell" to assist in development – proof of concept and implementation of the advanced intelligent algorithms for future "Smart Thermostats," such as, the proposed ALS model and zone controlled environment technique for Smart Homes and Buildings.

Hence, the simulator shall aid in investigation of the advantages of multi-zone versus single controlled house heating system; furthermore, to assist in proof of concept, development and implementation of ALS model (described in Chapter 6). Initial results showing the benefits of a controlled and programmable schedule achieved by a PCT versus the fixed (permanent hold) set points are presented, including the total energy consumption in KWh, and corresponding energy costs for several test cases. Additionally, the results of having a programmable weekly schedule based on the occupancy preferences and utility TOU rates, to optimize comfort and conservation of energy accordingly, are presented and further on elaborated in Chapter 6 (i.e., via "Observe, Learn and Adapt" algorithm, which is a practical implementation of the ALS model with the aid of a simulator described therein).

5.2 Model of a House Heating-Cooling System

The main building blocks of a house heating-cooling system model considered for the simulator, consists of the Outdoor Temperature Generator, Transducer, House Heating/ Cooling and a Thermostat unit, as shown in Figure 5.1. The simulator was implemented using C# high level language, by following the principles of object oriented design (OOD), and friendly user interface aspect of it. The complexity of a user interface in PCT is one of

the main reasons why most of the people do not utilize at their potential current PCTs [MEI08].

The purpose of Outside Temperature Generator is to read the outside temperature data from a comma separated value file and provide the hourly temperature data to the Thermostat. The Transducer computes the approximate heating losses of a house based on the outdoor and indoor temperature inputs and thermal isolation of a house model. House Heating Unit receives the inputs from a Thermostat and reacts by applying the control signal to turn on/off the heater (or air conditioner) and provides the heated/cooled air with the constant air flow and temperature to the house. Similar principles apply for a multi-stage heater/cooler units in a zone control environment, where multiple variables (instead of one) are needed to control the heater/cooler stages and/or adjust the heated/cooled air flow, based on the user preferred settings and/or requirements. The following subsection 5.2.1, elaborates on the multi-zone controlled environment concepts, which takes into account additional blocks and parameters of interest i.e. control of multiple heater/cooler stages and adjustment of the heated/cooled air flow rates in different zones.

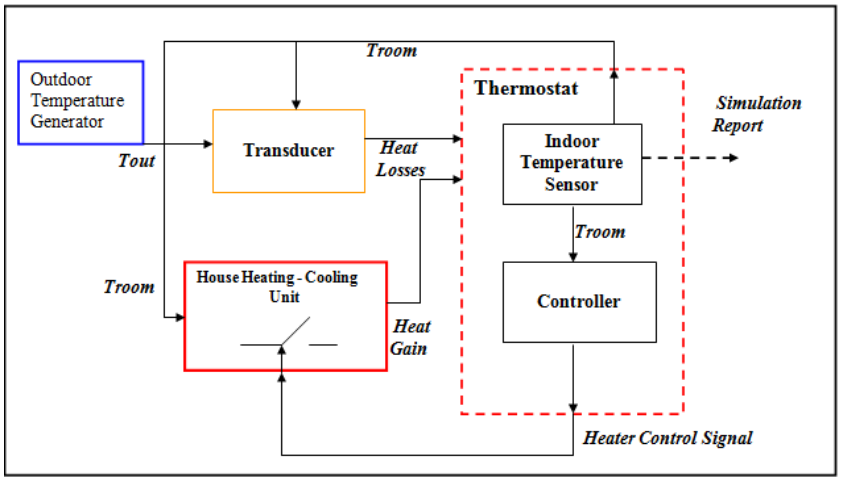


Figure 5.1 - Basic blocks of a house heating-cooling system

The Thermostat unit consists of the Indoor Temperature Sensor and a Controller. The Indoor Temperature Sensor receives the input from the Heating-Cooling System and Transducer, and computes the room temperature accordingly (taking into account the heat being generated by the heater and also the loss generated based on the house thermodynamics).

The Controller part of the Thermostat reacts when temperature values received from the Indoor Temperature Sensor are different than set point temperatures of a day.

5.2.1 Zone Controlled Environment for Smart Homes and Buildings

This section describes the proposed Knowledge-Base concepts applied in the simulator, which complement the ALS model for achieving a zone controlled environment in Smart Home and/or Building. The basic building blocks of the conceptual model used during design of the simulator were depicted in Figure 5.1. However, consideration for the zone controlled environment aspect, multiple heater/cooler stages of the HVAC units and air flow control, involves additional building blocks and parameters of interest to be considered.

Based on the energy savings incentives, time of the day, DR and TOU rates, desired/chosen set points and offsets, etc, one should consider control of the dampers within different zones of a house/building i.e. making use of sensors and/or actuators to adjust the heated/cooled air flow to different zones. Therefore, utilizing wireless sensors (i.e., monitoring) and actuators (i.e., variable air volume control) within the house and/or building are emphasized as the 'necessary ingredients' to complement the ALS model in achieving the optimal results.

A Knowledge-Base is created to receive input values for the air flow rate, heater temperature, and current room temperature. House simulator inquires the Knowledge-Base library, which after processing the information returns the recommended air flow rate, and it decides if only the first stage of heater should remain on, or if the second stage should be turned on or off. Whereas, air flow offsets and heat / cool offset, are variables which can be adjusted for best results based on the house thermal dynamics.

Notation:

A_f – airflow rate

H_t – heater

β_h – heater stages used

T_n - room temperatures

ΔT – temperature difference (gap between outside and inside temperature)

ρ_n – airflow offsets

μ_n - heater offset

K_b – Knowledge base function used to optimize zone control

Let $\beta_h = 2$; signifying HVAC system with two heater stages.

Let $\rho_{1,} \rho_{2...} \rho_n$ represent the available airflow offsets (which are controlled via actuators)

Let $K_b = f(A_f, H_t, \Delta T)$ represent a Knowledge-Base as function which relies on the airflow rate, heater and temperature difference within a specific zone.

Therefore, $K_b = f(A_f, H_t, \Delta T)$ in a generalized form is implemented as follows:

if $(\Delta T < T_1)$
$A_f = A_f \pm \rho_1$
$\beta_h = 1$

if $(\Delta T >= T_1$ *AND* $\Delta T < T_2)$
$A_f = A_f \pm \rho_2$
$\beta_h = 1$

if $(\Delta T >= T_2$ *AND* $\Delta T < T_3)$ (1.17)
$A_f = A_f \pm \rho_3$
…

if $(\Delta T >= T_k)$
$\beta_h = 2$
$H_t = H_t + \mu_n$
$A_f = A_f \pm \rho_{n-1}$

if $(\Delta T >= T_{n-2}$ *AND temperature* $< T_{n-1})$
$\beta_h = 2$
$H_t = H_t + \mu_n$
$A_f = A_f \pm \rho_n$

return A_f

The above Knowledge-Base rules depend mainly on the house parameters, such as house size (air volume), number of windows, estimated air leakage, ventilation, zones to control, etc. Therefore, it is considered as a part of expert system, as an applicable approach, which can be fine tuned for optimal performance, for any specific house and/or building.

Similar approach as depicted in (1.17) can be used for adjusting the daily temperature offsets based on the desired set points of a day/night and also based on the occupancy detection via sensors. Thus, zones which are not frequently used, can be optimized, such as to conserve energy, by keeping them at the maximum tolerable offsets (set by the user),

while the set point temperatures in living room and/or frequently used spaces are being kept at the comfortable settings.

The following subsection describes the thermal model of a house used, based on the fundamental principles of thermodynamics and [AME09] recommendations, from which simulation formulas are derived.

5.2.2 Thermal Model of a House for 'Simulation Engine'

Initial thermal model used in the 'simulation engine' is a simplified model of house heat and cool gain/loss system, in order to observe and analyze its response, and to enable planning for a feasible implementation of a 'Smart Thermostat'. Heat flow through a house depends on many factors, such as the difference in inside and outside temperature, conductivity of building materials, thickness of materials, etc. In order to provide better approximation to the real world scenarios, implementation of the model considers the contributing parameters in the overall heat loss, such as the ventilation losses and house air leakage. Heat transfer process (i.e., from the warmer side to the colder side) is affected by thermal resistivity of materials k, temperature difference Δt, wall thickness L and area A [AME09], as depicted below:

$$q = k\frac{(t_{s1} - t_{s2})A}{L} = \frac{(t_{s1} - t_{s2})}{\frac{L}{kA}} \qquad (1.18)$$

While the heated air supply into the house is represented as:

$$q_h = \Delta T \cdot A_{flow} \cdot c \qquad (1.19)$$

Where ΔT is the difference between indoor and outdoor temperature, A_{flow} is the heated airflow, and c is the specific air capacity.

Based on the equations above and principles of thermodynamics, the derived thermal model considered for the rate of heat/cool losses and gain of a house, is the following:

$$\left(\frac{dQ}{dt}\right)_{losses} = \frac{T_{room} - T_{external}}{R_{eq}} \qquad (1.20)$$

$$R_{eq} = \frac{l}{\lambda}$$

(1.21)

R_{eq} - Equivalent thermal resistance of the house $\left(\frac{m^2 K}{W}\right)$

l - Wall Thickness (m)

λ_{th} - Thermal conductivity $\left(\frac{W}{m \cdot K}\right)$; for air @ 20°C $\lambda = 0.0257 \left(\frac{W}{m \cdot K}\right)$

T_{room} - Room temperature (°C); this is temperature as read by the room sensors.

$T_{external}$ - External temperature (°C); this is the temperature read by outside sensor at the outside wall of the house.

The initial model of a heated air supply to the house is modeled as follows:

$$\left(\frac{dQ}{dt}\right)_{heat} = (T_{heat} - T_{room}) \cdot M_f \cdot c \qquad (1.22)$$

Where,

$\frac{dQ}{dt}$ - Heat flow from the heater into the room

M_f - Air mass flow rate through heater (kg/sec)

$\boldsymbol{c}$ – Specific heat capacity of the air at constant pressure (J/kg K)

$$\frac{dT_{room\,increment}}{dt} = \frac{1}{M_{air} \cdot c}\left[\left(\frac{dQ}{dt}\right)_{heat} - \left(\frac{dQ}{dt}\right)_{losses}\right] \qquad (1.22)$$

$\frac{dT_{room\,increment}}{dt}$ - Rate of temperature change inside the room

$M_{air} = d \cdot V$

M_{air} - Mass of air (kg)

V - Volume of the house (m^3)

Constants:

$\boldsymbol{c}$ = 1010 J/kg K

$\boldsymbol{d}$ = 1.21 kg/m^3

Temperature in Kelvin $T(K) = 273.15 + T(°C)$

When calculating $\left(\frac{dQ}{dt}\right)_{heat}$ T_{heat}, T_{cool} and T_{room} are converted from °C to K

$$Q_{heat} = \int_{t1}^{t2} \left(\frac{dQ}{dt}\right)_{heat} dt = M_f \cdot c \cdot (\Delta T_h \cdot t)/_{t1}^{t2} = M_f \cdot c \cdot \Delta T_h (t_2 - t_1)$$

$$Q_{losses} = \int_{t1}^{t2} \frac{T_{room} - T_{external}}{R_{eq}} dt = \frac{1}{R_{eq}} \cdot \Delta T_l \cdot t /_{t1}^{t2} = \frac{1}{R_{eq}} \cdot \Delta T_l \cdot (t_2 - t_1)$$

$$T_{room\_increment} = \frac{1}{M_{air} \cdot c}(Q_{heat} - Q_{losses})$$

$$Q_{cool} = \int_{t1}^{t2} \left(\frac{dQ}{dt}\right)_{cool} dt = M_f \cdot c \cdot (\Delta T_c \cdot t)/_{t1}^{t2} = M_f \cdot c \cdot \Delta T_c (t_2 - t_1)$$

$$Q_{cool\_losses} = \int_{t1}^{t2} \frac{T_{external} - T_{room}}{R_{eq}} dt = \frac{1}{R_{eq}} \cdot \Delta T_m \cdot t /_{t1}^{t2} = \frac{1}{R_{eq}} \cdot \Delta T_m \cdot (t_2 - t_1)$$

$$T_{room\_decrement} = \frac{1}{M_{air} \cdot c}(Q_{cool} - Q_{cool\_losses})$$

Where,

$\Delta T_h = T_{heat} - T_{room}$ $\Delta T_c = T_{room} - T_{cool}$ *t_1, t_2 – time: lower and upper bound of simulation time*

$\Delta T_l = T_{room} - T_{external}$ $\Delta T_m = T_{external} - T_{room}$

5.2.3 Estimated Equivalent Thermal Resistance of a House

$$R_{walls} = \frac{L_{walls}}{K_{walls} \cdot A_{walls}}$$

$$R_{windows} = \frac{L_{windows}}{K_{windows} \cdot A_{windows}} \tag{1.23}$$

$$R_{eq} = \frac{R_{walls} \cdot R_{windows}}{R_{walls} + R_{windows}}$$

R_{eq} - Equivalent thermal resistance of a house $\left(\frac{m^2 K}{W} \right)$

R_{wall} - Thermal resistance of a wall $\left(\frac{m^2 K}{W} \right)$

R_{window} - Thermal resistance of a window $\left(\frac{m^2 K}{W} \right)$

L_{wall} - Thickness of walls [m]

L_{window} - Thickness of windows [m]

K_{wall} - Thermal coefficient of a walls [W/mK]

K_{window} - Thermal coefficient of a windows [W/mK]

A_{wall} - Area of a walls [m^2]

A_{window} - Area of a windows [m^2]

A simplified *Heater-Cooler Prototype System* was also implemented in order to 'emulate' few of its potential features by experiment utilizing hardware and firmware. The proposed system was experimented utilizing PIC24F microcontroller development board suitable for this scheme, other additional components and integrated circuits. In addition to the hardware, for firmware development, Embedded C language for PICmicro® family from *Custom Computer Systems Inc.* (shortly CCS) was chosen. For further details refer to the Appendix B.

5.3 Simulator Design

The House Simulator is a discrete event simulator, designed with the following key concepts in mind:

- To be able to simulate house heating/cooling systems under different test scenarios.
- To be able to utilize it as an "expert system shell" for development of future advanced smart thermostat learning algorithms.
- To be simple, user friendly and beneficial for everyone that is enthusiastic to invest little effort to save money and energy, and most importantly to aid in initiatives leading towards a sustainable environment for future generations.

5.3.1 C# Programming Language

C# is a relatively new programming language, which inherits the best from both worlds 'C and C++'. It is a fully object oriented language, with extensive features available for development of Windows and Internet applications. Among some of the features of C# are also the highly expressive syntax, full support for classes, OOD principles, type-safe coding, memory management, and vast available resources (advanced code editor, debugger, and toolbox) at fingertips of the programmer. Indeed, C# is simple, elegant, type-safe and powerful programming language for development of Windows and Internet applications, by making use of .NET Framework (platform for the development, deployment and execution of distributed applications).

5.3.2 Graphical User Interface – Main Features

The Graphical User Interface (GUI) of a House Simulator is depicted in Figure 5.2, which captures the main elements of a simulator look and feel. The left hand side of the Figure 5.2 shows the indoor and outdoor temperature graphs. The outside temperature profiles are part of the Outdoor Temperature Generator – where the weather profiles are read by the simulator, and represent the real weather data taken from the Canada's National Climate Archive and/or user defined. The indoor temperature profile represents a dynamic response of the house heating/cooling system which takes into account factors such as the thermal model of a house, heating loss and gain.

On the right bottom side of the Figure 5.2 is the Thermostat interface, which displays the data as simulation progresses, such as the indoor/outdoor temperatures, total energy consumed (KWh), total cost ($), heat/cool set points at different stages of simulation, TOU Rates at different points of simulation and the mode of operation (Heat, Cool, Auto, Off). The right top corner of Fig.5.2 depicts a multi-zone layout of a typical two storey house.

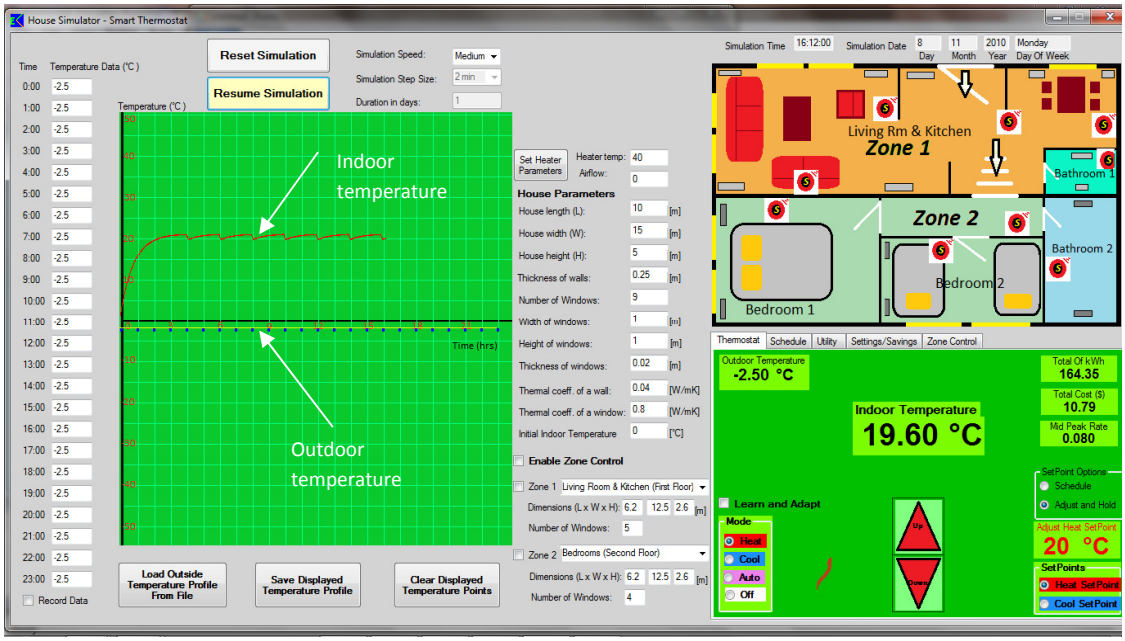


Figure 5.2 - Simulator graphical user interface

The user can also select different parameters of the house (dimensions, windows, walls, thermal coefficients, thickness of walls, windows etc.) and zones to simulate and analyze energy efficiency and heat losses that occur.

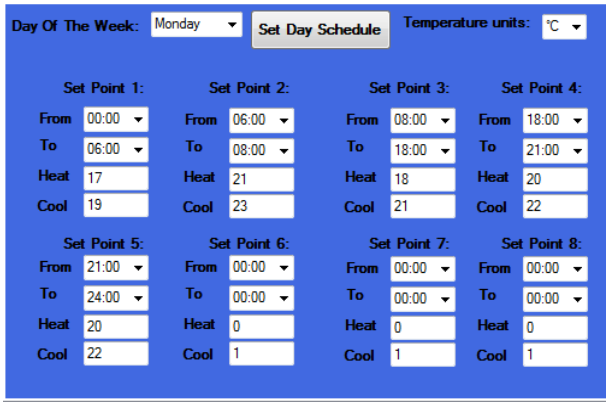


Figure 5.3 - Simulator schedule control

The simulator schedule control shown in Figure 5.3, enables one to select different daily schedules, including set points and time intervals during the simulation process (i.e. user can adjust heat and cool set points at different times of the day based on his/her preferences and working schedule).

The Figure 5.4 depicts the simulator DR and TOU rates control, which enables one to Opt-In and/or Opt-Out (i.e. participation) in utility DR events, thus enabling cost reduction incentives for the user, and helping utilities in peak load curtailments. Independently, the user can also enter different TOU Rates (typically set by the utilities: On Peak, Mid Peak and Off Peak) and observe its impact on monthly costs for different simulation scenarios.

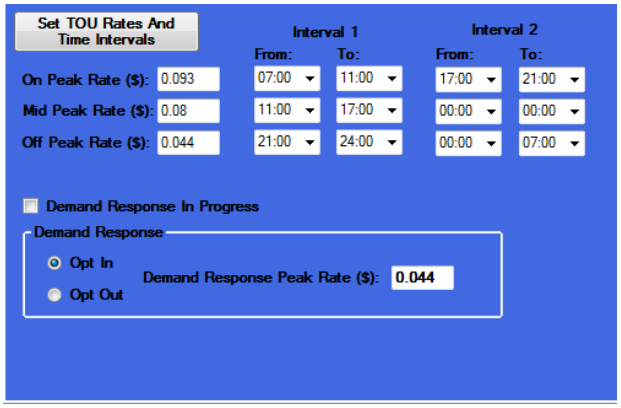


Figure 5.4 - Simulator DR and TOU rates control

Based on the selected values one can use different house parameters during simulation. The user can also select to simulate the house for different number of days, adjust the time intervals for any possible simulation scenario. Further improvements of the simulator and

additional controls are presented in Chapter 6 since they are closely related to the proposed OLA algorithm, which is a practical implementation of the ALS model.

5.4 Simulator Model

As described earlier in Section 5.2, heat flow through a house depends on many factors, such as the difference in inside and outside temperature, conductivity of building materials, thickness of materials, etc. Initial thermal model used in the 'simulation engine' mimics a house heat and cool gain/losses in order to observe and analyze its response; it enables for a feasible implementation of a 'Smart Thermostat' and furthermore implementation of practical adaptive learning system strategies (i.e. OLA).

The simulator conceptual model depicted in Figure 5.5 signifies the main conceptual blocks of the simulator model, such as the "House Simulator," "Smart Thermostat" and a placeholder for "Adaptive Learning" models to be implemented. Thus, taking into account the feasibility and ensuring that the placeholders -'hooks' for the ALS models within the simulator environment exist, prior to their implementation. The flow of information is from the "House Simulator" to "Smart Thermostat" and vice versa, similarly the same applies for "Adaptive Learning," which can be incorporated into the overall design, and affect the "House Simulator" and "Smart Thermostat" interchangeably.

The simulator conceptual model was translated and closely adapted into C# classes as depicted in Figure 5.6; indeed, as it will be seen at later stages of research in Chapter 6, implementation strategy is essential for the effective proof of concepts (i.e. typically, designs which do not account for scalability might become obsolete later on).

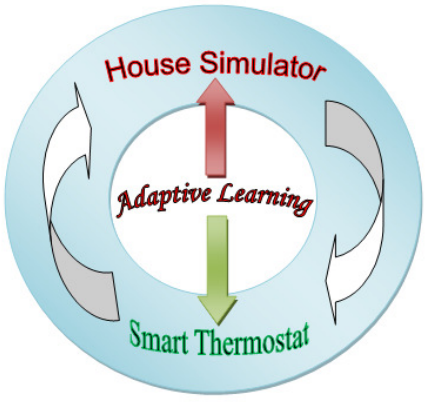


Figure 5.5 - Simulator conceptual model

The class Thermostat represents the main class of the simulator, where the control algorithm resides, while classes House, Room, Schedule and Utility are the other classes which represent a 'has-a' relationship with a main class, and are instantiated within main application with default system values, and properties. These values can be adjusted for different house system models and parameters. The main advantage in the implementation of the following design strategy is to be able to change any class parameters on the fly (i.e. simulation step size, interval, initial conditions, schedules, etc.). Enabling a flexible house model for experimentation and simulation under different conditions; furthermore, to be utilized as an 'expert system shell,' to explore and implement new adaptive learning system models.

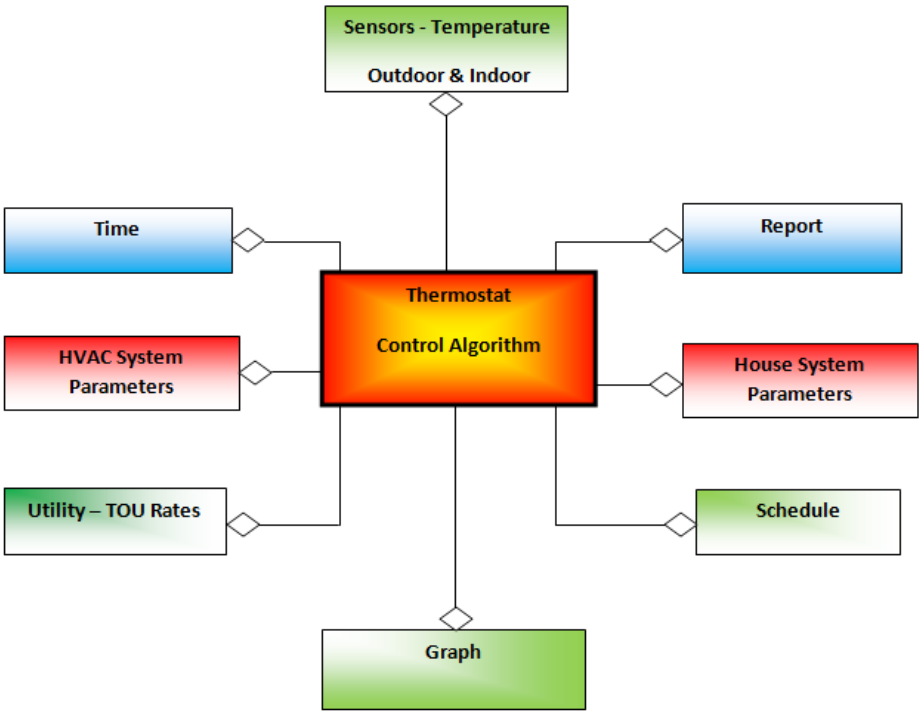


Figure 5.6 - Diagram of a simulator model

Indoor and Outdoor Temperature - provides information about the indoor and outdoor temperatures to the simulator.

Time **-** provides the necessary info for the simulation process, including simulation step size, speed, time of the day, set points at different instances of a day, etc.

HVAC System Parameters - react based on the heater and cooler state, temperature profiles, heating gain/loss, and other house system parameters of the design.

Utility / TOU Rates - provides the electricity cost at different times of day based on the TOU rates set by the Utility (e.g. On Peak, Mid Peak and Off Peak).

Graph - Plots 24 hour daily outside temperatures from the weather data; plots the indoor room temperatures at different time intervals (based on granularity of simulation step size selected i.e. 12 sec, 1 min, 2 min, 5 min, etc.).

Schedule – enables user to select and set the desired heat and cool set points for different times of day (hour: min) for each week day and weekends.

House System Parameters - enables the user to select different house parameters, including house volume, wall and window area, wall and window thickness and thermal coefficients, number of windows, etc.

Report – For each step of simulation, data is saved into a log file, which includes heat loss, heat gain, total energy consumption, energy consumption for different TOU rates, total cost for each step and the total cost for the entire simulation period as shown in table below.

Table 5.1 - Report data

Hour	*Minutes*	*External Temperature (C)*	*Room Temperature (C)*	*Heat Set Point (C)*
Cool Set Point (C)	*Heat Generated (J)*	*Energy Loss (J)*	*Total Heat (J)*	*Cost of Electricity (TOU Rates)*
Heat in kWh (TOU Rate On Peak)	*Heat in kWh (TOU Rate Mid Peak)*	*Heat in kWh (TOU Rate Off Peak)*	*Total kWh Cost (On Peak active)*	*Total kWh Cost (Mid Peak active)*
Total kWh Cost (Off Peak active)	*Total kWh Cost ($)*			

Thermostat & Control Algorithm – provides the main control of the heating/cooling system, by orchestrating the input output relationships, calculating the essential values based on the parameters supplied to the simulator (i.e. equivalent thermal resistance of a house, heat/cool set points, air flow rate, heater capabilities, etc). Improvement and prevalent features of adaptive learning for Smart Thermostat, such as use of OLA and Knowledge-Base shall be introduced and discussed in the subsequent chapter.

5.5 Performance Results

The TOU Rates are based on the projected rates that are in effect since 2010, from the Hydro One website, (On Peak rate = 0.093$, Mid Peak rate = 0.08$ and Off Peak rate = 0.044$). The set point (SP) values for typical schedule used are based on the Tables 5.2 and 5.3.

Table 5.2 - Monday to Friday schedule

Time of Day	Heat Set Point (°C)	Cool Set Point(°C)	Description
00:00 to 6:00	17	19	Sleep
6:00 to 8:00	21	23	Wake
8:00 to 18:00	18	21	Away
18:00 to 24:00	20	22	Home

Table 5.3 - Saturday and Sunday schedule

Time of Day	Heat Set Point (°C)	Cool Set Point(°C)	Description
00:00 to 8:00	17	19	Sleep
8:00 to 12:00	21	23	Wake
12:00 to 18:00	18	21	Away
18:00 to 24:00	20	22	Home

The default house parameters used during the simulation are shown in Table 5.4.

Table 5.4 - Initial house parameters

House Parameters	Value	Unit
House length	15	m
House height	10	m
House width	5	m
Number of windows	8	
Windows length	1	m
Windows width	1	m
Thermal coeff. Of a wall	0.04	W/mK
Thermal coeff. Of a window	0.8	W/mK
Thickness of walls	0.25	m
Thickness of windows	0.02	m
Initial room temperature	0	°C

The response time of a system based on the parameters indicated on the Table 5.4, for different average output temperatures (0 °C, -2.5 °C and -5 °C) and set point of 20 °C are shown in Figures 5.7 and 5.8, whereas the initial inside house temperature is 0 °C.

The response time of a system depicted in figure 5.7, to reach the desired set point based on an average outside temperatures of -5 °C, - 2.5 °C and 0 °C is approximately 175 min,

130 min and 105 minutes, respectively. In this case, the actual set point has an offset of ± 0.5 °C, however due to the step size (min on/off time of HVAC system set at 5 minutes) the average offset i.e. dead band of the system is approximately ± 1°C.

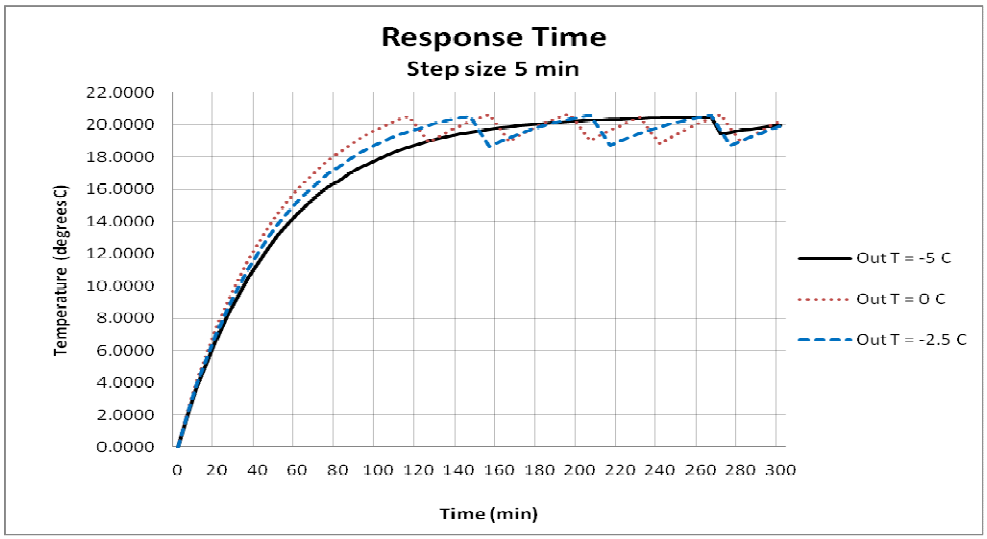


Figure 5.7 - Response time of a system (step size 5 min)

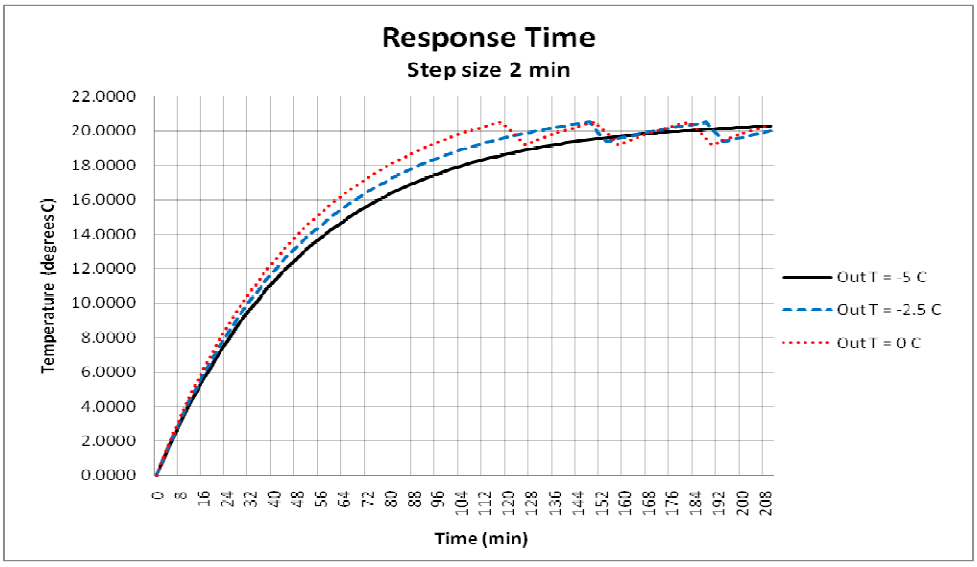


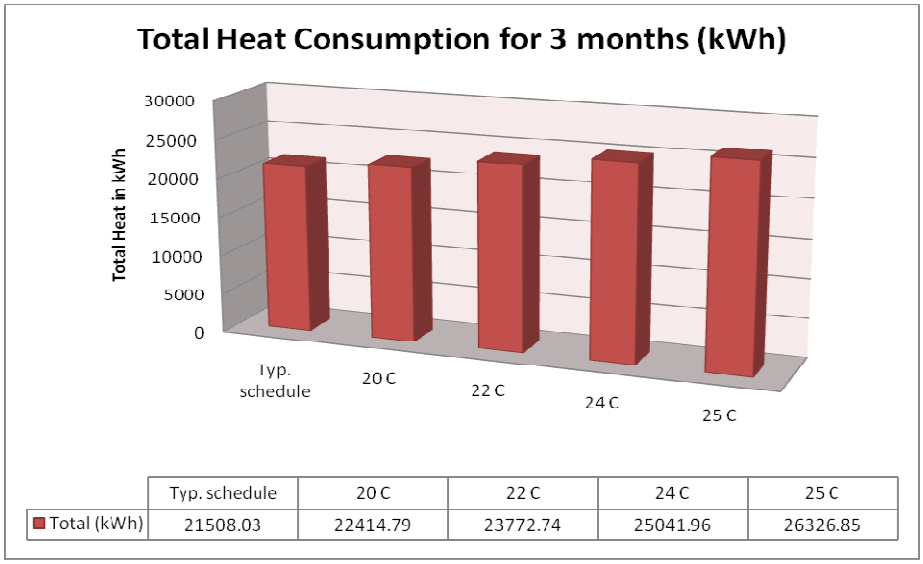
Figure 5.8 - Response time of a system (step size 2 min)

In Figure 5.8, response time of system - settling time is similar to the one shown in Figure 5.7. Based on identical conditions i.e. an average outside temperatures of -5 °C, - 2.5 °C and 0 °C set point is reached after approximately 180 min, 130 min and 108 minutes,

respectively. The actual set point has an offset of ± 0.5°C, similar to the previous case. However, due to better granularity of a step size (min on/off time of HVAC system set at 2 minutes) the average offset i.e. dead band of the system is maintained within ± 0.54 °C.

The Figure 5.9 depicts the simulation results of a house for the duration of three months during the winter season (weather data for outdoor temperatures, used for simulation is from the Canada's National Climate Archive for month of December 2008, January and February 2009). The actual simulator model was compared to the HOT 2000 Simulator [NAT08]. Thus, for an average indoor temperature set point of 20 °C, analogous house parameters and initial conditions, monthly report was compared against a similar case of the proposed simulator (considering an average outdoor temperature of 0°C); whereas the monthly margin of the calculated error for the estimated energy consumption was 7.57%.

The simulation results of total energy consumption for house heating in KWh (duration of 3 months) are shown in Figure 5.9. Respectively, in Figure 5.10 total associated costs in dollars for the energy consumed, are shown. In both Figures 5.9 and 5.10, the first column shows the consumption for three months (KWh or dollars spent), based on a typical weekly schedule of set points. While, the second to fifth column show the consumption (KWh or dollars spent) for fixed set points during the entire period of 3 months (i.e. set point of 20 °C, 22 °C, 24 °C and 25 °C).



	Typ. schedule	20 C	22 C	24 C	25 C
Total (kWh)	21508.03	22414.79	23772.74	25041.96	26326.85

Figure 5.9 - Total heat consumption for 3 months (KWh)

The potential savings in energy consumption and cost is possible to achieve with only 4 different set points weekly schedule preferences, instead of fixed/permanent set points. Taking as an example the permanent set point of 20 degrees C for 3 months versus the typical schedule, one would observe that the cost associated with keeping a heat set point of 20 degrees C for 3 months is $1374.1, while typical schedule total costs are $1319.49, savings of $54.6 (equates to 906 KWh energy savings).

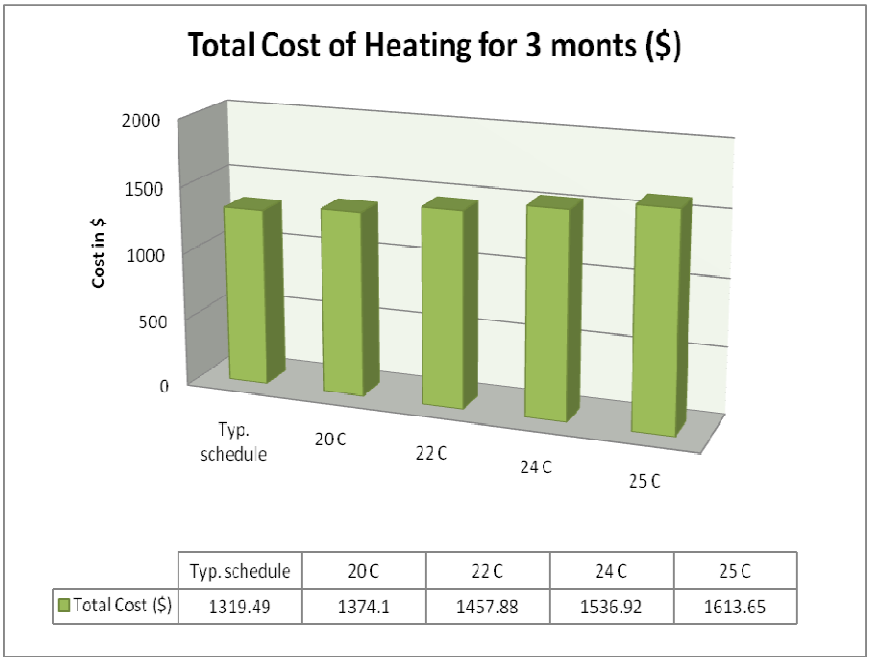


Figure 5.10 - Total cost of heating for 3 months in dollars

In addition, Figure 5.11 depicts the results of the simulation for a zone controlled house environment, which reflect a better yield with respect to energy conservation; hence, more cost savings for a zone controlled house. While, in Figure 5.12, simulation results of energy costs demonstrate the energy consumed and TOU rates, during 3 month period of heating.

The effect of a zone controlled environment can be considered very important step towards better energy conservation and management. The role of sensor nodes in this case is obvious necessity for any "Smart Thermostat," which would be used to control the indoor temperatures in a zone controlled environment. Furthermore, it can be said that in addition to the possible lack of intelligence in current PCTs, another important factor which impacts their performance, is when PCTs are equipped with only one or two sensors i.e. typically for

an intelligent PCT, more than a few sensor/actuator nodes are needed for an optimal control of a multi-zone controlled environment.

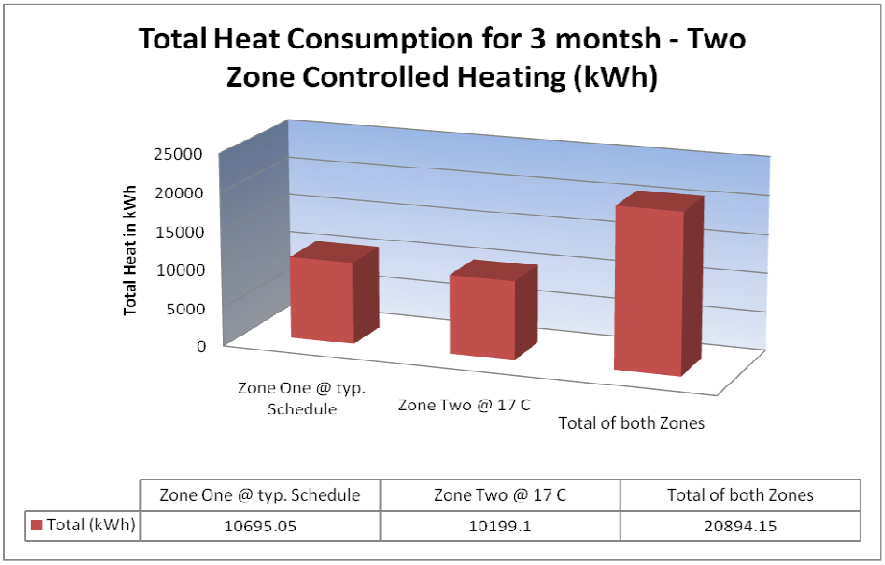


Figure 5.11 - Total heat consumption of a two zone system

From the simulation results shown in Figure 5.11 (energy consumption) and Figure 5.12 (associated costs), it can be observed that the zone control environment yields better energy efficiency and cost savings results, when compared to a single zone controlled environment.

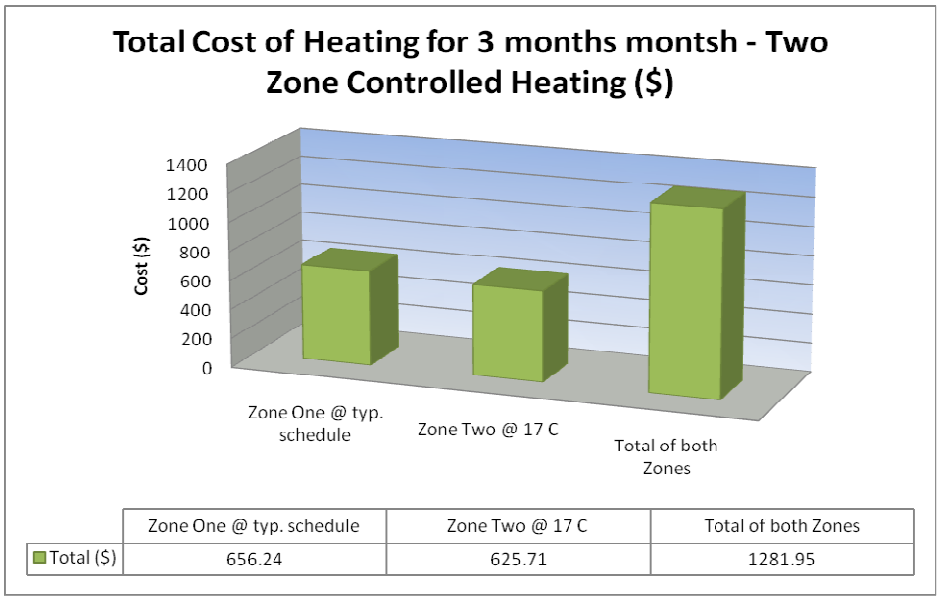
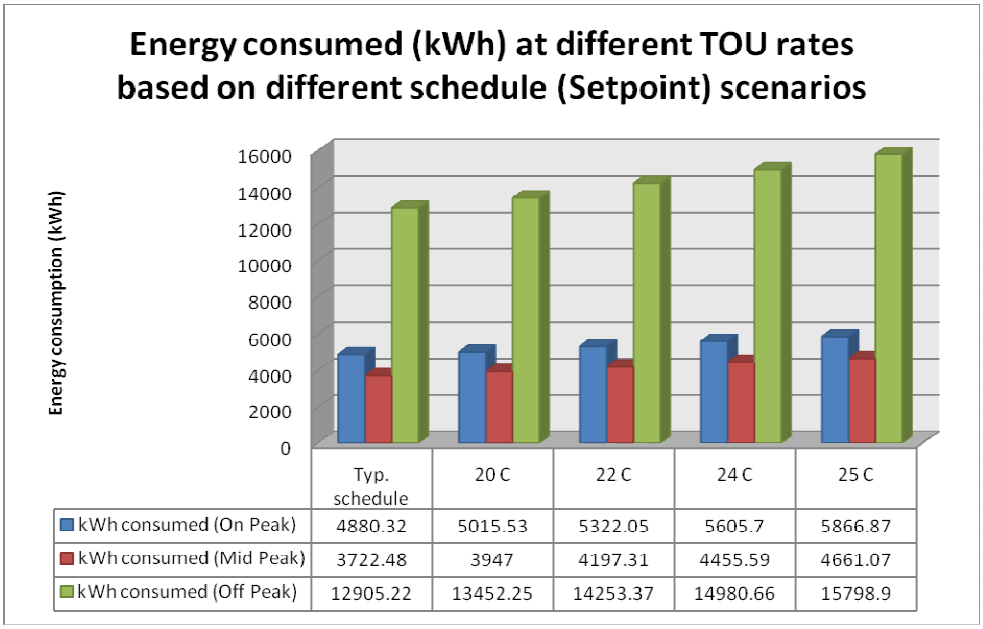


Figure 5.12 - Total cost of heating of a two zone system



	Typ. schedule	20 C	22 C	24 C	25 C
kWh consumed (On Peak)	4880.32	5015.53	5322.05	5605.7	5866.87
kWh consumed (Mid Peak)	3722.48	3947	4197.31	4455.59	4661.07
kWh consumed (Off Peak)	12905.22	13452.25	14253.37	14980.66	15798.9

Figure 5.13 - Energy consumed based on different TOU Rates (KWh)

Figure 5.13 shows the energy consumption for simulation of a typical weekly schedule of set points and permanent set points of 20 °C, 22 °C, 24 °C and 25 °C, based on the different TOU rates applied (at different times of day) for a period of 3 months.

An indication of TOU rates, comfort and consumption portrays the main concepts which are leading to incentives proposed by EGUs, and are being endorsed by many customers around the globe. Hence, the effect of an adaptable thermostat could prove to be beneficial into balancing the effect of higher costs with the comfort of the user, minimizing the energy consumption, whilst maintaining the user comfort.

5.6 Summary

The outcome results from the simulation of a house heating system, indicate possible energy conservations by utilizing a PCT with flexible schedule(s), and furthermore by zone controlled systems. The energy and cost savings of a house heating system for a period of three months were demonstrated by analyzing different test scenarios. The disadvantages of current PCTs are their user interface and lack of intelligence, which if enhanced, indeed could lead us one incremental step forward towards a more efficient energy conservation and comfort in our living environments. However, it has to be noted that the schedules and patterns change, and in most cases users utilize permanent hold set points instead of using

the existing schedules due to many factors, including PCTs complexity to program, frequent schedule changes, etc. Last but not least, one of the main objectives for enhanced performance and learning capabilities of PCTs is also the minimal possible user intervention, whilst not jeopardizing the user comfort. Thus, an intelligent PCT would prove to be more satisfactory for consumers, too. In order to attest beneficial for further energy savings, cost reduction and sustainable resources, PCTs can be complex systems, however, must be "smart and adaptable devices," with a simple user interface.

In addition to its usefulness for simulation of a house heating/cooling system under different scenarios, more importantly the simulator was built to be used as a tool to assist in implementation and further development of the intelligent algorithms (learning strategies) for future PCTs i.e. "Smart Thermostats."

Therefore, the simulator conceptual model described herein, which consists of a "House Simulator," "Smart Thermostat," with a 'hook' for the "Adaptive Learning" algorithm, facilitates the efforts during the implementation phase (i.e. practical ALS model implementation). Thus, as depicted previously in Section 5.4, Figure 5.5, the "House Simulator" and "Smart Thermostat" are two integral parts of the simulator, which interact with each other (same is in a real-world scenario i.e. HVAC and PCT in a house environment), and allow integration of supplementary functionality, components/objects, and/or new algorithms.

Moreover, the interaction of building blocks within a simulator, provide a good ground for evaluation of different scenarios, as shown in this chapter. Few extra features closely related to the practical ALS model implementation are described in the Chapter 6.

CHAPTER 6

ADAPTIVE LEARNING SYSTEM IMPLEMENTATION

6.1 Introduction

The leading edge technology in the area of intelligent systems and smart sensor networks, are an essential part of our everyday life; their evolution will help us to better utilize our resources (energy saving initiatives) and enhance our way of living. The role of PCT is to provide consumer with a means to manage and reduce energy use, whilst accommodating their every day schedules, preferences and needs. It has to be noted that the use of only one AI model, would not always suffice to bring forward the best systemic solutions; hence, the utilization of different techniques has lead to the materialization of many hybrid intelligent systems (which combine at least two intelligent technologies).

The concepts of an adaptable PCT herein are being investigated in order to aid in bringing forward systemic solutions which are adaptable, 'energy aware' and easy to use. The Observe Learn and Adapt (OLA) algorithm proposed and described in this chapter, illustrates the actual implementation of the proposed ALS model; an integration of wireless sensors and artificial intelligence concepts towards the same objective: adding more intelligence to a PCT for better energy management and conservation in Smart Homes and Buildings. Moreover, as described earlier in Chapter 5, a simulator tool was designed in order to be used as an "expert system shell" to assist in development, implementation and verification of ALS model via OLA. In addition, the actual performance results of learning and adaptability of a PCT equipped with OLA, as a result of the occupant's changing schedules and/or patterns are shown. Additionally, the overall system improvements with respect to energy consumption and savings are demonstrated via simulation for the zone controlled home equipped with OLA and Knowledge-Base, versus a home without zone control, Knowledge-Base nor OLA.

6.2 Description of the OLA Algorithm

6.2.1 Overview

OLA in essence is a 'reflection' of hybrid intelligent system concepts, combining rules-based expert system, unsupervised learning approaches, and wireless sensors, in order to bring forward a new algorithm which can be used to add learning capacity to the existing PCTs; 'transforming' current PCT into a 'Smart Thermostat' concept - a PCT with adaptable learning capabilities.

In a 'nutshell' OLA behavior is as follows: observe via sensors, acquire and learn from the new inputs (by comparing new knowledge with the existing one), and adapt outputs (via actuators) or other, based on the decision made (if decision is true, adds new clustering knowledge to the dedicated cluster group, i.e. see Daily Clusters). The main building blocks of the OLA algorithm for 'Smart Thermostats' are depicted in Figure 6.1 below.

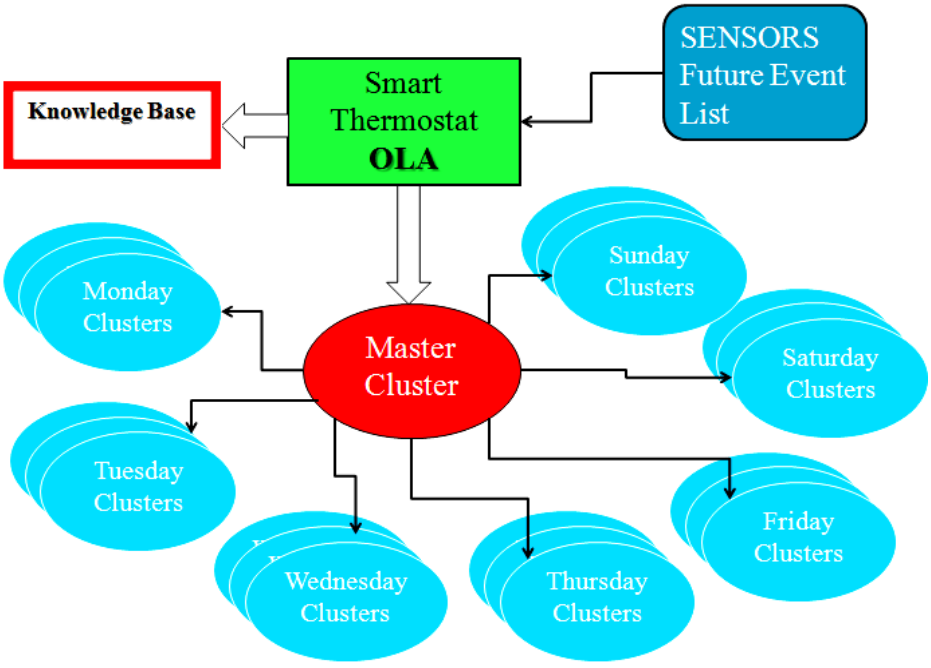


Figure 6.1 - Conceptual diagram of the elements used in OLA

Knowledge-Base - is an additional knowledge to the 'Smart Thermostat' which contains information about the tuned settings and weight options, necessary for optimality in case of the entire house or zone control scenarios. In Figure 6.1, the wide arrow from OLA to Knowledge-Base and to the Master Cluster, indicate that OLA can request and retrieve information from the Knowledge-Base and Master cluster (and its Daily Clusters). Whilst the narrow arrow from the Future Event List (FEL) to the OLA, indicate that FEL is an input to OLA. The Knowledge-Base is the existing knowledge implemented within OLA.

Master Cluster - contains information for every existing cluster and includes also additional information necessary for adapting the best solution at any time. Further information related to the structure of Master Cluster structure is given in Appendix C.1.

Daily Clusters - contain specific information related to the user patterns, schedule and activity for each day of a week (updates knowledge when required). Further information related to the structure of Daily Cluster structure is given in Appendix C.2.

Future Event List (FEL) - emulates actual sensory/actuator data: "Sense via sensors, Act via actuators." Contains information about future events and is used to emulate different patterns and also to verify the learning and adaptability of the algorithm at work. Few FEL parameters used emulate wireless sensors functions (occupancy detection, temperature readings at different spaces, actuation); hence, wireless sensors play a major role in a smart home application of OLA. On the other hand, actuators are the key contributors used for adjustment of the air flow (variable air valves) in different zones and/or rooms. Although, wireless and wired sensor can measure the same physical properties, feasibility of wireless sensors is their ease of integration in the existing building environments, where wired sensors are not preferred due to the wiring. Furthermore, in cases when relocation of sensors in necessary within a house and/or a building, wireless sensors offer greater flexibility, whilst wired sensors would pose a challenge. On the other hand, actuators are the key contributors used for adjustment of the air flow rates in different zones and/or rooms.

6.2.2 Main Steps of the Algorithm

Figure 6.2 depict the main implementation steps of the OLA algorithm based on the proposed ALS model (illustrated and described in Chapter 3, Section 3.2.1). Each block has an associated number based on which additional description is given below, while in the subsequent Section 6.2.3 the main routines are explained.

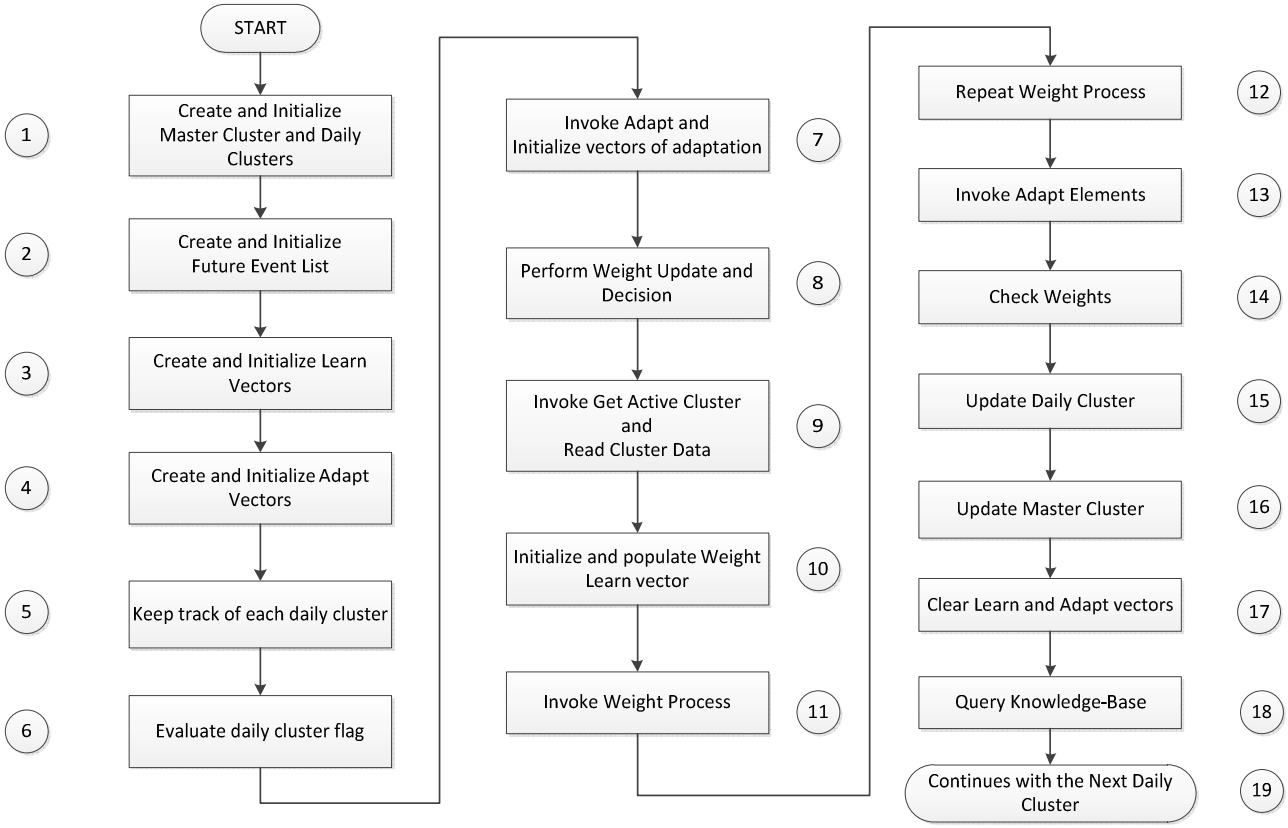


Figure 6.2 - Main steps of OLA algorithm

1. Create and initialize Master and Daily clusters with default parameters.
2. Create and initialize FEL.
3. For each day of a week, create and initialize Learn vectors (Learn vectors extract information from the wireless sensors i.e. FEL).
4. For each day of a week, create and initialize Adapt vectors based on the Learn vector inputs. Adapt vectors add adaptability parameter status (value 0 for unchanged and 1 for changed).

5. Keep track of each daily cluster by ensuring not to overlap existing data elements (three consecutive weeks) with the new ones, when populating the vectors.
6. Evaluate daily cluster *'adaptWeek'* flag (based on the 'adaptWeek' value invoke proper Daily Cluster Adapt method).
7. Invoke *'Adapt'* and initialize vectors of adaptation. Decide which particular elements for a period greater than two weeks need to adapt (from Adapt vectors).
8. Perform *'Weight Update and Decision'* based on the active daily cluster (day of a week).
9. Invoke *'Get Active Cluster'* of the day and *'Read Cluster'* data.
10. Initialize and populate *'Weight Learn'* vectors (with cluster data from Learn vector for particular week only (not the entire set).
11. Invoke *'Weight Process'*. Compare each element of weight decision vectors. Based on the initial set of tolerances for elements of the vectors, decide which weight to assign to each one.
12. Repeat the *'Weight Process'* - by invocation of a method for all subsets of data (i.e. since limit for learning and adaptability is set to perform adaptation after three occurrences of the daily cluster).
13. Invoke *'Adapt Elements,'* which takes as an input day of a week and potential vectors to be adapted.
14. Invoke *'Check Weights,'* which takes as an input the element to be analyzed, day of a week and the associated daily cluster vector. '*Check Weights'* makes a decision based on a set of rules according to the weights of daily cluster vectors. It returns the adapted value (after evaluating the vector elements and the associated weights) for each element of the cluster.
15. *Update Daily Cluster* (Monday, Tuesday, etc.) based on the adapted vector values, and currently active cluster).
16. *Update Master Cluster* knowledge base – reads and adapts Master cluster with new cluster updated information, related to daily cluster updated.
17. Clear Learn and Adapt vectors.

18. When processing temperature data, query the Knowledge-Base for optimal settings, and if necessary adjust the amount of heat, and/or offsets into the dedicated zones of the smart home.
19. Continue with the next Daily Cluster.

6.2.3 Main Routines of the Algorithm

The main routines of OLA algorithm are explained below.

Initialize FEL

Reads FEL file and creates a new FEL object which contains read data to emulate sensory inputs. Thus, it has different occupant's patterns and/or schedule changes for the simulation process, in order to emulate different scenarios. Structure of FEL file is as shown in Table 6.1 below.

Table 6.1 - FEL structure

Future Event List Structure		
simDay	Set Point #2 Cool	Zone 2 Heat offset Day
simMonth	start time	Zone 2 Heat offset Night
simYear	end time	Zone 2 Cool offset Day
sWeekDay	Set Point #3 Heat	Zone 2 Cool offset Night
Event start time	Set Point #3 Cool	Settings of Savings
Event ends time	start time	Learn and Adapt schedule
Nr. of persons in Zone1	end time	Opt In
From (specific time of day)	Set Point #4 Heat	DR Event
To (specific time of day)	Set Point #4 Cool	from
Activity of persons in Zone1	start time	to
Nr. of persons in Zone2	end time	TOU High Rate
From (specific time of day)	Set Point #5 Heat	from
To (specific time of day)	Set Point #5 Cool	to
Activity level of persons in Zone2	start time	TOU Medium Rate
Heat Set Point#1	end time	from
Cool Set Point#1	Zone 1 Heat offset Day	to
start time	Zone 1 Heat offset Night	TOU Low Rate
end time	Zone 1 Cool offset Day	from
SetPoint #2 Heat	Zone 1 Cool offset Night	to

Populate Learn Vectors

Learn vectors are populated from the FEL information and have a size of 120 elements (40 elements * 3 weeks, representing data available for 3 consecutive daily cluster occurrences). The Learn vectors contain information based on the schedule of a day and have the following structure:

{Heat SetPoint, Cool SetPoint, Start time, Stop time, Heat weight, Cool weight, Start time weight, Stop time weight}; where the vector elements in positions 0 to 3 are Heat Set Point, Cool Set Point, Start and Stop time of Set Points, while in positions from 4 to 7 are their respective weights (associated with elements from 0 to 3). The adapted schedule for 'Smart Thermostat' has five Set Points (SP): *{SP1, SP2, SP3, SP4, SP5}*. Therefore, we have in total: 5 SP multiplied by 8 elements in each, which equals to 40 elements for each occurrence of the 'specific daily schedule', and 120 for a total of three occurrences. If the data elements are not changed, value of -1 is entered for that particular element of the Learn vector.

Populate Adapt Vectors

Adapt vectors are populated only after comparison with the Learn vector data and adaptWeekDay indicator (takes note of occurrences of particular day of a week i.e. adaptWeekMon for Monday, adaptWeekTue for Tuesday, etc. and based on which occurrence of the day it is (first, second or third) it retrieves correct set of elements from Learn vector.

Adapt vector structure is as follows: {Heat Set Point, Cool Set Point, Start Time, Stop Time}. Adapt vector is populated only after comparison with the Learn vector data. The adaptWeekDay indicator takes note of the occurrences of a particular day of a week i.e. adaptWeekMon for Monday, adaptWeekTue for Tuesday, etc. and based on the occurrence of a day (first, second or third) it retrieves the correct set of elements from the Learn vectors. Furthermore, for each element and particular daily occurrence (first, second or third) of the Learn vector, it performs the steps as depicted in Figure 6.3.

Adapt vector does not keep track of weights associated with each element of the Learn vector; hence it has only 20 elements for each daily occurrence. Its structure is {Heat SetPoint, Cool SetPoint, Start Time, Stop Time} * five different SetPoints for the entire active daily cluster.

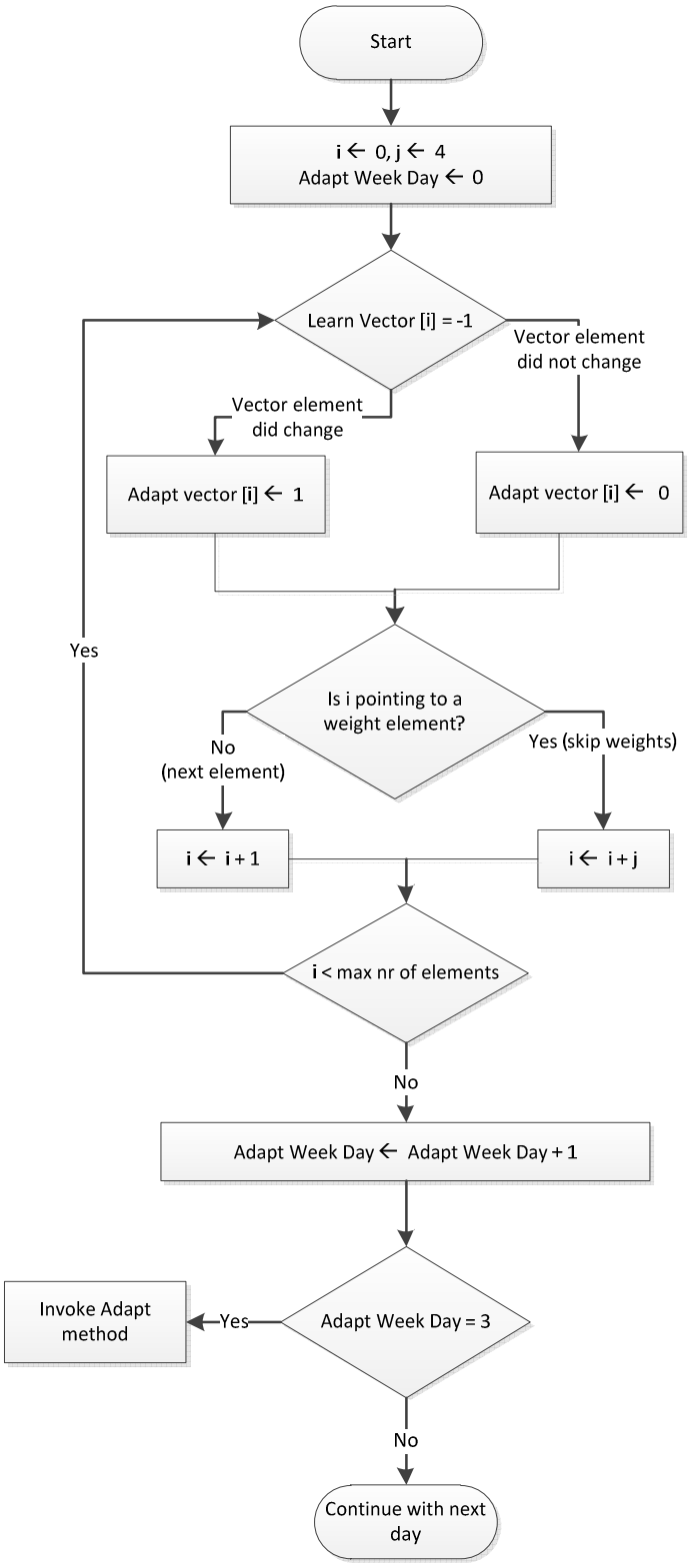


Figure 6.3 - Populate adapt vectors

The term "i←i+j" where j = 4, shown in Figure 6.3 indicates that the weights are skipped when populating the adapt vectors. Thus, learn vectors, in addition to the Heat SP, Cool SP, Start time and Stop time, contain values of the weights associated with each element, While, the adapt vectors contain only the values to be adapted. The *adaptWeekDay* set to a value of 3 (i.e. before the *Adapt* method is invoked) is based on several assumptions. For example, considering a scenario when the occupant changes temporarily the heat set point temperature, perhaps in an extremely cold winter day; this might be one time occurrence, and not necessarily the preferred 'permanent set point' temperature. Therefore, prior to adapting to the occupant's preferred set points, OLA compares at least 3 instances of the occurrence, before adapting to a change.

Similarly, a PCT schedule has typically five different set points during a day, which implies that at least five different start and stop time intervals are available (refer to Table 2 "Time of Day" column) where potential changes might occur. Therefore, in cases when the occupant leaves a house one hour earlier than usual, this might be only one time occurrence, and not, a typical permanent schedule or pattern change. Thus, the term *adaptWeekDa*y = 3 (refer to Figure 6.3) does not imply that three weeks are essential for OLA to learn. Instead, it indicates that three occurrences of a change are considered as a common factor for the adaptation of new values. It has to be noted that OLA still reacts and applies optimal settings for savings, when house is unoccupied. In addition, OLA will keep track of the occupant's pattern changes and it adapts (without user interaction) if the similar pattern is repeated.

Adapt

Adapt routine has input parameter the day of the week. Hence it is triggered for each day (1 for Monday, 2 for Tue, etc.). Three temporary vectors with a limit of 20 elements each are initialized (v1, v2, v3) and each one is populated with the data from Adapt vector (v1 for the first occurrence, v2 for the second occurrence, and v3 for the third occurrence of a specific day). For each element and particular day occurrence, verification is done to see if the change is consistent or not. After which, the elements associated with the temporary vector

v_t are copied to a permanently assigned vectorToAdapt of a day (vectorToAdapt1 for Monday, vectorToAdapt2 for Tuesday, etc.). Figure 6.4 depicts the main steps of the routine.

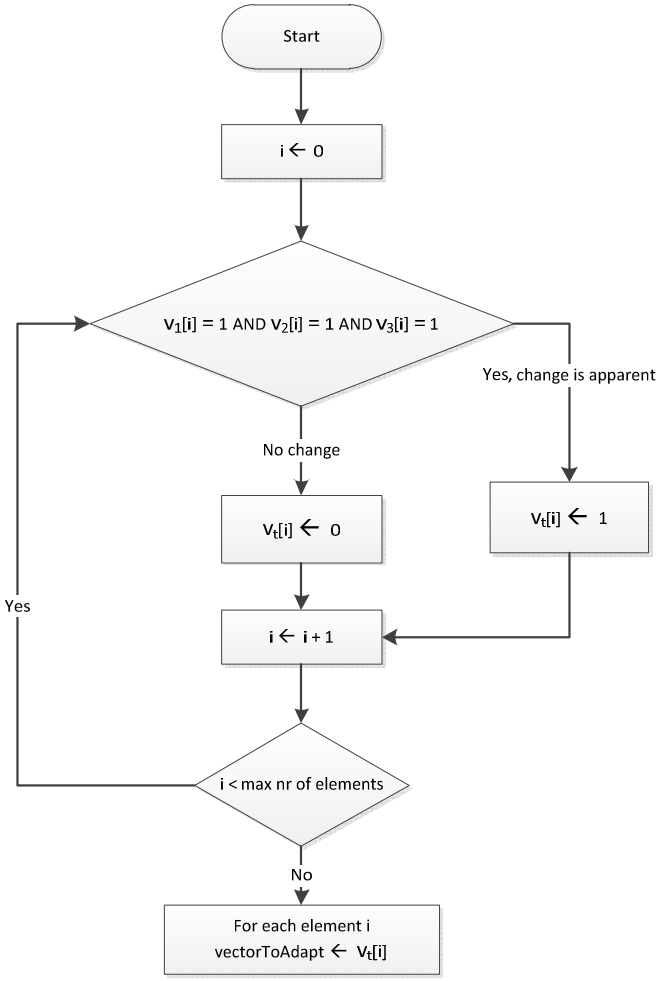


Figure 6.4 - Adapt routine

Weight Update Decision

Consists of three main routines as described below under a, b and c.

a) ***Read Cluster Data***

Reads the active daily cluster file; adds all data in a *ReadData* array list for further processing. Whereas all the necessary data is copied into a SPWeightDecision vector (size of 40 elements).

Afterwards, another SPWeightLearn vector extracts the data from the appropriate Learn vector of the daily cluster (new data from FEL) for comparison with the SPWeightDecision vector (existing data from the active daily cluster).

b) Get Active Cluster of Day

Based on the day of a week, it acquires the daily cluster active cluster file (information which is updated each time Master Cluster) and returns the cluster file name to the *Read Cluster Data*. Next, it performs the above processing for all the elements of interest (Heat Set Point, Cool Set Point, Start and Stop times; for all Set Points defined in cluster data). And for each *Weight Process* completed (based on the day of the week) it assigns the data and weights to a Monday vector, Tuesday vector, etc. Whereas, HEAT_TOL = 1, COOL_TOL = 1, START_TIME_TOL = 1 hr, STOP_TIME_TOL = 1 hr, are values that can be adjusted, and are available for further tuning.

c) Weight Process

First it initializes the weights to zero. Thereafter, for each week (first, second and third) it performs a weight process, whereas all the data elements from the SPWeightDecision and SPWeightLearn vector are compared, as depicted in Figure 6.5.

Adapt Elements

Adapt Elements routine, consists of four main sub-routine as described below under a, b, c and d.

a) Check Weights (int elem, int day, double [] vec)

Checks each element of a vector, of the specific day consisting of {Heat SetPoint, Cool SetPoint, Start Time, Stop Time, WeightHeat, WeightCool, WeightStart, WeightStop}, for five SetPoints of a day. From ALS model description we know that β is a weight multiplier coefficient which can be adapted based on the problem at hand. In case of ALS implementation via OLA, $\beta = 1$ and $\lambda = 0.1$.

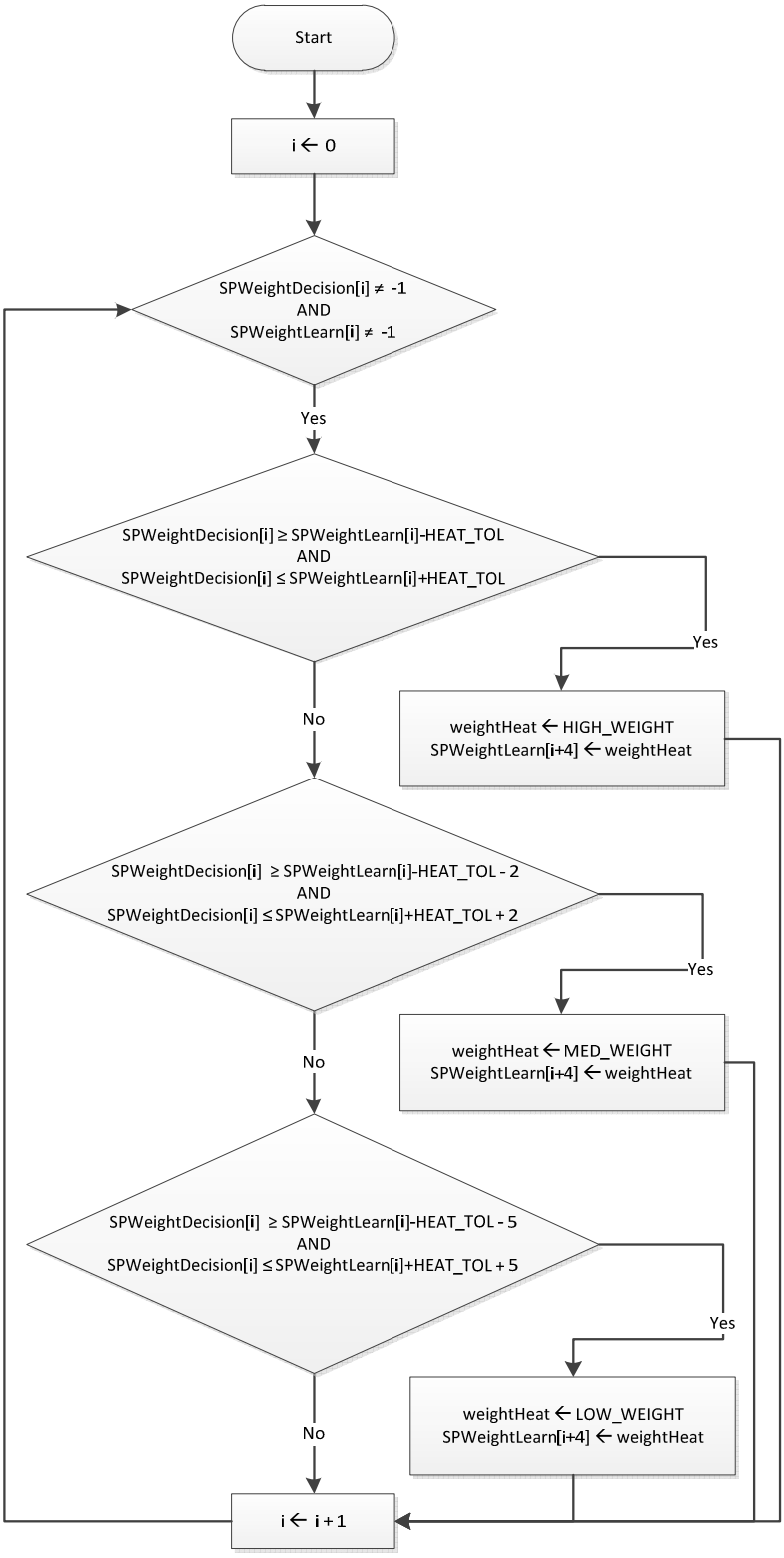


Figure 6.5 - Weight process

Hence, there are three different weight: HIGH_WEIGHT = 0.1, MED_WEIGHT = 0.05 and LOW_WEIGHT = 0.025, which can be assigned to any daily vector based on the proximity of the actual value to the particular element for three consecutive occurences of a particular day. The rule-based decision according to weights is based on the following possible combinations:
LLL, LLH, LHL, LHH, HLL, HLH, HHL, HHH, MMM, MMH, MHM, MHH, HMM, HMH, HHM, LLM, LML, LMM, MLL, MLM, MML, LMH, LHM, MLH, MHL, HML, HLM. Where H stands for HIGH_WEIGHT, M for MED_WEIGHT and L for LOW_WEIGHT.

The resulting value returned from the *CheckWeights* is based on the of weight occurences, i.e. if all three weekly occurences have the same weights, average of three weekly elements is returned; if only two weekly occurences of the particular vector elements have high weights while third one has low weight, the resulting value returned is based on the avarage of data elements corresponding to high weights, ignoring the low weight element. The low weight signifies a major shift from typical existing value, hence the approach is slightly conservative and tends not to make radical changes to the existing schedule or set points. Hence, in cases such as when major shifts from the existing schedules and/or setpoints occur, adaptation will take place only after three consecutive occurences of the low weights.

b) Update SetPoints

Each time that *CheckWeights* routine is invoked, returned result is assigned to the appropriate element of the daily cluster schedule (when all the elements are updated the schedule is updated and becomes active).

c) Update Master Cluster

Invokes *Read Cluster Data* routine and it updates the master cluster active file parameter and file number that was replaced by current file. Master cluster structure is shown in Appendix C.1.

d) Update Daily Clusters

Following the previous step, *Update Daily Cluster* routine creates new cluster based on the existing and new knowledge which was adapted (i.e. updated Set Points, Start and Stop times, or other parameters of interest). Daily Cluster Structure is shown in Appendix C.2.

Knowledge-Base

A sample method of how Knowledge base is used to adjust the air flow rate and to turn on/off different heater stages of HVAC (for different indoor and outdoor temperatures) is the *Airflow Rate and Heater Adjustment* depicted in Figure 6.6.

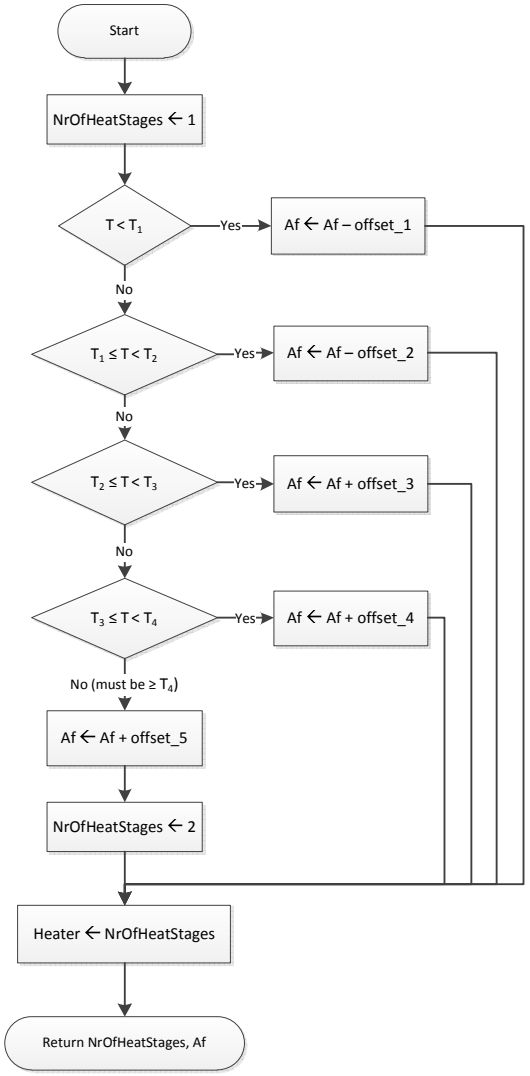


Figure 6.6 - Airflow rate and heater adjustment

Each time the processing of temperature data from the 'Smart Thermostat' is performed, rules-based expert system checks for the optimal settings, and the acquired facts are used to by the system, in order to adjust the amount of heat, air flow rate and/or offsets into the dedicated zones of the smart home

Airflow Rate and Heater Adjustment is a sample method from Knowledge-Base class, which receives as input, values for the air flow rate, heater temperature, and current room temperature and inquires the Knowledge-Base library. The Knowledge-Base after processing the information, based on the algorithm depicted in Figure 6.6, it returns the recommended air flow rate, and the necessary active heater stages to be turned on or off. Whereas, the *offset_0,…,offset_5* are differnt airflow rates (incremental) and *NrOfHeatStages* are the actual number of heater stages (both are variables which can be adjusted for best results based on the house thermal dynamics). While, T represent the difference between indoor and outdoor temperatures, and T_1, T_2, T_3, T_4 and T_5 are the limits (i.e. T_1 = 10 °C, T2 = 20 °C, T_3 = 30 °C and T_4 = 40 °C).

Knowledge-Base after processing the information returns the recommended air flow rate, and if the second stage of the heater should be turned on or not.

6.3 Application of OLA and the Big Picture

Most of the current PCTs are equipped with capabilities to communicate with a Smart Meter (a two way communication device, capable to monitor power consumption, communicate to Utility and/or other end devices, such as PCTs, home appliances, load switches, etc.).

OLA can be used in current PCTs (leading to 'Smart Thermostats') in order to augment their performance, by introducing devices capable to learn and adapt (while offering optimal comfort and conservation) in a Smart Home environment. Furthermore, 'Smart Thermostats' are meant to be used as devices which do not require constant programming input by the user.

In order to provide the most efficient savings, and help manage the peak load demand; a multi-sensor system must be capable to communicate with the core controller unit (i.e. 'Smart Thermostat'), process inputs (TOU rates, DR) to and/or from the Utility to the core controller unit or dedicated end sensor/actuator nodes. Since typically wireless sensors and/or actuators have limited memory and/or processing power, synergy of the

reinforcement learning and agent-based techniques such as described in subsequent chapter can be used to further enhance the performance of the overall system by enabling the sensor nodes to act as 'intelligent agents' and extract the possible future peak load events after receiving the potential functions and time of a day interval - from Utility (via Smart Meter).

The aim of OLA is to optimize the comfort with respect to energy consumption by learning occupancy preferences and patterns, thus enabling the comfort zone adjustment, including the individual control of heating/cooling of rooms, and/or the entire home.

6.4 Improvements of the Simulator

Few new concepts were added and updated within the simulator, during the implementation of OLA, including Knowledge-Base and FEL (i.e., sensors) and controls, such as the Settings for Comfort and Savings and Offsets for Zone Controlled Environment, as depicted in Figure 6.7 below.

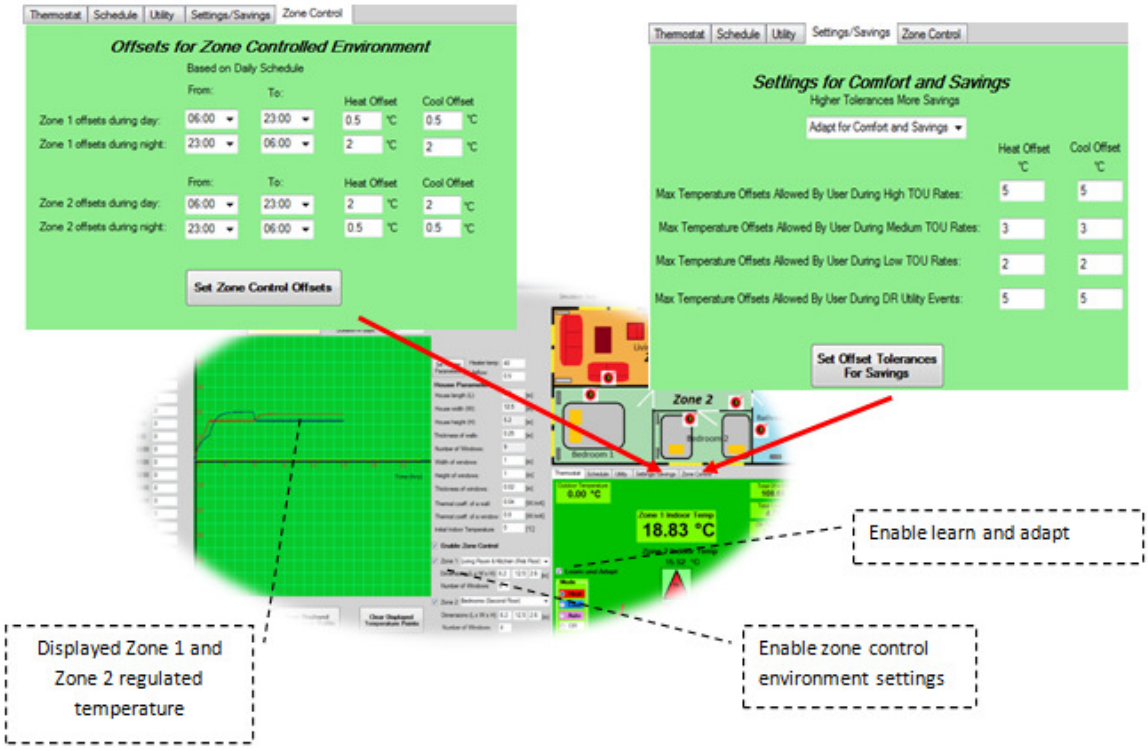


Figure 6.7 - Improvements of the simulator

Knowledge-Base and FEL, as described in previous section, were implemented as an integral part of OLA algorithm and added to the Simulator. Similarly, 'Settings for Comfort and Savings' control enables the user to select the desired maximum and minimum

temperature offsets during DR events and/or high, medium and low TOU rates, while 'Offsets for Zone Controlled Environment' control, enables user to select the desired zone offsets during particular times of day (i.e. set different offsets for different zones during day and night time). The above settings are also captured within the FEL and Daily Cluster structure. In addition, the displayed zone 1 and 2 data simultaneously, and the options to enable full house control versus zone controlled via OLA enabled algorithm, aids in overall performance verification of a system.

6.5 Performance Results

Several simulation scenarios are considered in order to demonstrate the performance and verification of OLA at work. The initial settings used for the scheduled set points and their daily intervals, for week days and weekends are depicted in Tables 6.2 and 6.3 below. Whilst, the main house parameters used for the simulation scenarios are shown in Table 6.4.

Table 6.2 - Monday to Friday schedule

Time of Day	Heat Set Point (°C)	Cool Set Point(°C)	Description
00:00 to 6:00	17	19	Sleep
6:00 to 8:00	21	23	Wake
8:00 to 18:00	18	21	Away
18:00 to 24:00	20	22	Home

Table 6.3 - Saturday and Sunday schedule

Time of Day	Heat Set Point (°C)	Cool Set Point(°C)	Description
00:00 to 8:00	17	19	Sleep
8:00 to 12:00	21	23	Wake
12:00 to 18:00	18	21	Away
18:00 to 24:00	20	22	Home

The TOU rates used are the rates based on the projected rates that are taking effect in 2010, from the Hydro One website, (On Peak rate = 0.093$, Mid Peak rate = 0.08$ and Off Peak rate = 0.044$). The initial TOU rate intervals used for TOU Rates during the simulation are shown in Table 5.5. The TOU rates applied during the weekends are assumed to be Off Peak.

Table 6.4 - House parameters

House Parameters	Value	Units
Length	6.2	m
Width	12.5	m
Height	5.2	m
Thickness of walls	0.25	m
Nr of windows	9	
Width of windows	1	m
Height of windows	1	m
Thickness of windows	0.02	m
Thermal coefficient of a wall	0.04	W/mK
Thermal coefficient of a window	0.8	W/mK
Initial Indoor temperature	0	°C

Table 6.5 - TOU rates

TOU RATE	From	To	From	To
On Peak	7:00	11:00	17:00	21:00
Mid Peak	11:00	17:00	N/A	N/A
Off Peak	0:00	7:00	21:00	24:00

A sample scenario of the updated simulator at work, is shown below in Figure 6.8, which shows two different simulation scenarios of the entire house and zone controlled house with OLA enabled. The zone controlled environment displays zone one (shown in red) and zone two (shown in blue) temperatures. Based on the activity of persons in zones, OLA decides which offset tolerances to apply (i.e., typically, if a zone is sensed by wireless sensors as unoccupied the offset tolerance is greater, whilst if zone is occupied, the offset tolerances are smaller).

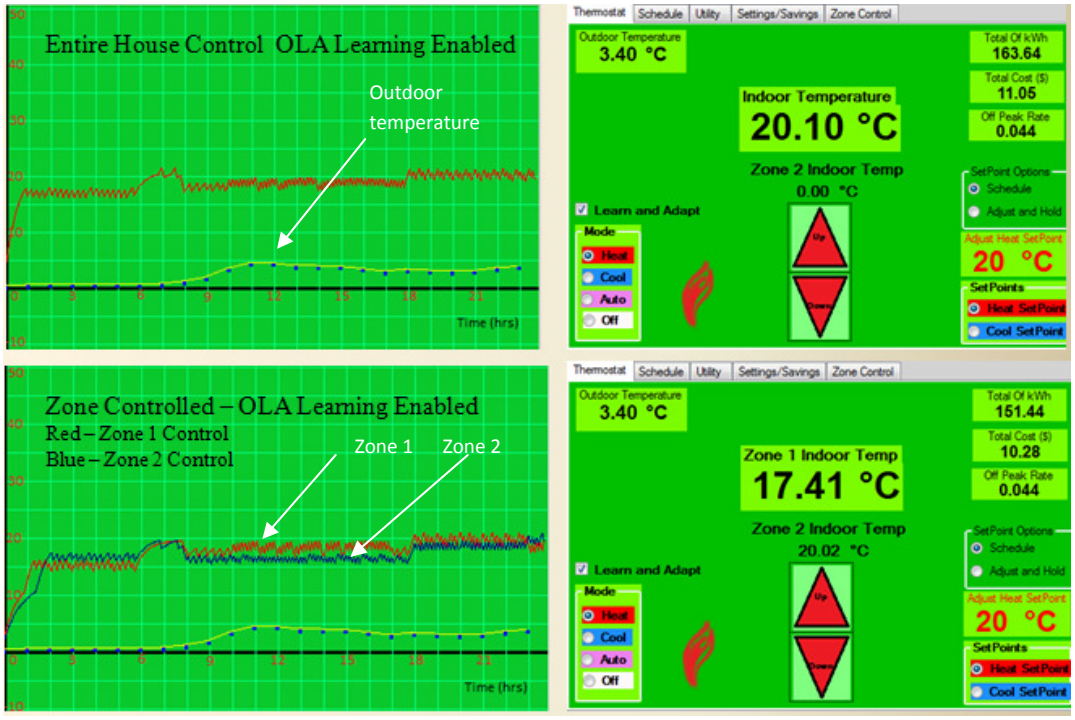


Figure 6.8 - Simulation (entire house and zone controlled)

The first case considered for verification of functionality was with and without learning enabled (no changes to the existing scheduled set points or intervals). The average outside temperature set for this case was 0 °C for the duration thirty days. The results of simulation are shown in Table 6.6. From the results, it can be observed that potential savings in case of entire house with OLA enabled versus the entire house with OLA disabled are 79 KWh.

Table 6.6 - Results of OLA (Entire house vs. zone controlled)

Configuration	OLA	Period (days)	Energy consumption (kWh)
Entire house	Disabled	30	5224
Entire house	Enabled	30	5145
Zone controlled	Enabled	30	4945

Furthermore, when comparing zone controlled house with OLA enabled versus the entire house with OLA disabled, potential savings equate to 279 KWh. Hence, improvements with respect to relative energy consumption with and without algorithm at work are apparent.

In order to further observe and verify the functionality of the algorithm, a 30 day simulation was run with and without the preconfigured FEL (i.e. with and without sensors) as inputs to OLA; weather data for outdoor temperatures used in this case was from the Canada's National Climate Archive (with an average of – 7.06 °C). The configuration was set to zone controlled environment with OLA enabled (and FEL not active) for cases 2 to 5.

The first case depicts energy consumption when the configuration is set for the simulation of entire house when OLA and Zone Control (ZC) are disabled. Recorded results indicate estimated energy savings of 147 KWh when Zone Control and OLA are enabled. The learning and adaptation period of the algorithm is set to three weeks. The different cases of OLA with and without FEL changes (to be detected by algorithm) are depicted in Table 6.7. Algorithm adapts to a new change in the schedule and/or user pattern, after the third occurrence of a specific change in user pattern and/or schedule.

Table 6.7 - Results of OLA (with and without FEL/Sensors)

Case Nr.	ZC/OLA	FEL Sensors	Changes to be detected	Energy consumption (kWh)
1	No/No	No	N/A	7194
2	Yes/Yes	No	N/A	7047
3	Yes/Yes	Yes	Mon/Tue Set Points 2 and 3	7042
4	Yes/Yes	Yes	Mon/Tue/Wed/Thur/Fri Set Points 2 and 3	7036
5	Yes/Yes	Yes	Mon/Tue/Wed/Thur/Fri Set Points 2, 3 and 4	7034

For case study number 3, changes initiated/detected by FEL/Sensors and adapted by OLA are for Monday Cluster daily patterns, whereas the end time for Set Point 2 and start time for Set Point 3 changed from 6:00 to 8:00 AM, to 6 to 7:30 AM, respectively for Tuesday Cluster from 6:00 to 8:00 AM, to 6:00 to 7:15 AM. The changes take effect during the last week of simulation only (i.e. first three weeks are dedicated to learning the patterns). The actual changes taking effect were observed in the Master Cluster and Monday/Tuesday Cluster files (although cumbersome, main idea why input/output file system method was used to implement Master and Daily Cluster objects, instead of arrays, is to be able to verify and analyze the changes only when program is running in debug mode, but also when the simulation process terminates); the changes taking effect, and the occurring total energy differences/savings due to the changes, validate the OLA successful adaptation to the change.

The outcome of adapted and/or updated schedule for two concurrencies (only for two days of a month) of different patterns are reflected in the results of case study number 3 in comparison to case 2 (zone controlled heating with OLA enabled but FEL inactive); where the estimated energy savings are 5 KWh (and indeed are 152 KWh in comparison to case 1).

Furthermore, case study number 4 reflect the adapted changes for Monday to Friday scheduled end time of Set Points 2 and start time of Set Point 3 as follows:
Monday Cluster Set Points 2 and 3 changed from 6:00 to 8:00 AM, to 6:00 to 7:30 AM. Tuesday set points 2 and 3 changed from 6:00 to 8:00 AM, to 6:00 to 7:15 AM. Wednesday Cluster Set Point 2 and 3 changed from 6:00 to 8:00 AM, to 6:00 to 7:00 AM. Respectively, Thursday Cluster Set Points 2 and 3 changed from 6:00 to 8:00 AM, to 6:00 to 7:00 AM, while Friday Cluster from 6:00 to 8:00 AM, to 6:00 to 7:45 AM. The estimated energy saving of 11 KWh in comparison to reference case 2 are confirmed (since in this case the

occurrence of newly adapted patterns from the 'Smart Thermostat' takes place in the last week of the simulation process). In addition to the case 4 scenario described before, case 5 introduces two additional parameter changes; the additions are: Monday Cluster Heat Set Point 3 changed from 18 °C to 17 °C, and end time of Set Point 3 and start time of Set Point 4 from 7:00 AM to 18:00 PM, to 7:30 AM to 18:30 PM.

As before, the changes take place after three weeks of OLA learn and adapt schedule. The estimated savings of the daily adapted patters (only for the last week of simulation process) indicate 13 KWh in comparison to the reference case study 2.

The significance of quantitative results (when multiplied by 4.2 as an approximate measure of monthly occurrences) shows savings of 54.6 KWh per month for very small changes in pattern behavior of the user; in fact the total conservations add up to a total of 54.6 + 147 KWh = 201.6 KWh per month for OLA enabled, ZC enabled and FEL active. Hence, adapting to changes makes a difference in the overall results, and demonstrates performance and benefits of OLA and ZC enabled Smart Home with FEL/Sensors active.

Table 6.8 - Statistics of set points start/stop times

Pattern changes - Set Points start / end times in hours					
	Mon	**Tue**	**Wed**	**Thur**	**Fri**
1st week	7.5	7	7.25	9.5	7.5
2nd week	7.75	6.75	8.5	9.75	10.75
3rd week	8.25	6.5	8.75	9.5	7.5
Mean	7.83	6.75	8.17	9.58	8.58
Std deviation	0.38	0.25	0.80	0.14	1.88
Conf. int. (95%)	**(7.45, 8.22)**	**(6.5, 7)**	**(7.36,8.97)**	**(9.44, 9.73)**	**6.71,10.46)**
OLA adapted values	**7.75**	**6.75**	**8**	**9.5**	**7.5**

The statistical analyses of multiple tests performed in order to validate the OLA algorithm at work are shown in Tables 6.8 and 6.9. Two scenarios shown in Tables 6.8 and 6.9 emulate pattern changes of user preferences, with respect to the Set Point Start and End times and Heat Set Points, for each day of the week.

Table 6.9 - Statistics of heat set points

Pattern changes – Heat Set Points in Celsius					
	Mon	Tue	Wed	Thur	Fri
First week	17	19	22	20	19
Second week	21	23	19	18	18
Third week	19	21	17	23	22
Sample mean	19	21	19.33	20.33	19.67
Std dev.	2	2	2.52	2.52	2.08
Conf. int. (95%)	17,21	19,23	16.82,21.85	17.82,22.85	17.59,21.75
OLA adapted values	19	20	18	19	18.5

As presented in Tables 6.8 and 6.9, different input parameters were used for each day of the week (for both cases). The outcomes of OLA adapted values (after three weeks of learning) are normally distributed and fall within the limits of the 95 percent confidence interval of the sample mean values (for confidence interval calculation refer to Appendix D).

From Table 6.8, we can observe that the OLA adapted values, for example, for the Monday schedule when the occupant leaves a house at 7:30 AM (converted to 7.5 hours in simulator) first week, and on the second and third week leaves at 7:45 AM and 8:15AM, respectively. The OLA adapted value after the third occurrence is not the average value of three consecutive occurrences which is 7:49:48 AM (i.e. 7.83 hours), instead is 7:45:00 AM (i.e. 7.75 hours). Thus, implies that OLA adapts the value which is closer to frequent occurrences. Furthermore, referring to Table 8, from Friday`s occupant pattern changes (time of leaving the house), it can be observed that OLA adapted value is 7:45:00 AM (7.75 hours) which, in fact is the occurrence of pattern on first and third week). Hence, OLA did not adapt the average value of three weekly occurrences, which is 8:34:48 AM (8.58 hours) affected by a change of pattern on second week of observation 10:45 AM (7.75 hours). Instead it adapted the value which is closer to the typical user patterns observed during the first and third occurrence, and at the same time leads to energy conservation (if the occupant typically leaves a house at 7:45 AM, there is no need to keep the indoor temperature at 21°C, for an extra hour, knowing that scheduled leave heat set point, set by the occupant is 17 °C). The same principle is also valid for the heat set points scenario as depicted in Table 6.9. By taking the week of Friday as an example, where the first week's '*Away'* set point (refer to

Table 6.2, Monday to Friday schedule) was changed from 18 °C to 19 °C, on second week it was set to 18 °C, and it was changed again to 22 °C in the third week. If the average of three occurrences was taken, the adapted value would have been 19.67 °C, which indicates a shift of almost two degrees from a typical schedule initially chosen by the occupant. Thus, the OLA adapted value of 18.5°C, is closer to the occupant's scheduled set point, and it leads to energy conservation.

In all the scenarios described above, the relations of energy savings to the occupant's comfort are in agreement with the occupant's preferences of initial schedule, temperature limits and tolerances, based on which OLA acts. One of the advantages of OLA enabled house is that the comfort of the occupant is not altered during the process. Instead, occupant's preferences are maintained, while energy savings are achieved.

6.6 Summary

The OLA algorithm enables 'Smart Thermostat' to utilizes FEL (i.e., sensors) and learning capabilities to adjust and adapt schedule based on user input pattern changes and preferences, taking into consideration parameters, such as zone offset tolerances, comfort settings, utility DR and TOU rates, etc.

OLA encompasses the main attributes which could provide potential improvement to the current PCTs with regard to their lack of intelligence, by adding learning capabilities and flexibility to act and adapt without intervention due to the occupant's pattern and/or schedule changes (as it was demonstrated in this chapter). The notion of "Smart and Adaptable Devices," has to be considered as an important attribute of current PCTs. OLA with the aid of sensors and application of ALS model learning technique captures the essence of an actual PCT reflecting into a smart and adaptable device. As it was demonstrated during the performance evaluation, the OLA adapted values (after three weeks of training) are obtained and are within 95 percent of the sample mean values. Moreover, performance results indicate potential savings with respect to energy consumption for zone controlled OLA enabled house versus a house with OLA disabled and no zone controlled environment.

CHAPTER 7

CONCLUSIONS AND FUTURE RESEARCH

7.1 Conclusions

The main objective of this thesis was to investigate and bring forward an '*Adaptive Systemic Solution*,' with the aim of improving energy management and conservation in Smart Homes and Buildings, and by the same token enhancing the learning capabilities of current PCTs - '*transforming*' them into smart and adaptable devices i.e. 'Smart Thermostats'.

Section 1.7 of Chapter 1, briefly summarizes thesis contributions, whilst below are described the concluding remarks of the thesis contributions, and research efforts.

An Adaptable Hybrid Intelligent System utilizing WSN and AI techniques was proposed, based on which, a novel ALS model utilizing WSN, rule-based system and ART concepts, was proposed and demonstrated. The proposed ALS analytical model (described in detail in Chapter 3) is a technique which enables PCTs to learn and adapt to the occupant's input pattern changes and/or other parameters of interest. In order to verify the ALS model, two different scenarios involving pattern changes of temperature set point of the PCT, including leave time, were considered. In both cases, the ALS model adapted values were much closer to the occupant's preferred values, and demonstrated improvement with respect to energy conservation, when compared to just averaging of values.

A new algorithm for finding global maximum of a function with minimal function evaluation/iterations in a predefined interval within a two dimensional space was proposed (Chapter 4). The proposed algorithm represents a synergy of concepts from the RL and agent-based techniques, for use in small-scale embedded systems with limited memory and/or processing power, such as wireless sensor/actuator nodes (i.e., intelligent agents). The "Reinforcement Learning and Agent-based Search" application was implemented in order to observe the algorithm at work and demonstrate its main features. The main benefits of the proposed algorithm are to 'bridge a gap' between the EGUs (i.e., utilities) and "Smart Thermostats" into "Smart Grid" initiatives. Several sample functions with different

characteristics were tested in order to verify the performance of the algorithm with 'RL only' versus 'RL and Agent-based Search' - emulating different scenarios, such as finding critical global and local peak load demand points, which can be used in conjunction with the utilities (via DR and TOU rate incentives) in managing peak load demands. During the verification process of the algorithm at work for function*: $y = 2\ ((\cos(\pi x) / x) + 1)$* where $x \in [1, 5]$, step size of 0.005 and tolerance of 0.05, it was observed that for the 'RL and Agent-based Search' technique global and local maximum was found after 30.8 iterations (calculated from 10 trials) versus 189 for 'RL Only, while the number of function evaluations was found to be 109.2 versus 382 for 'RL Only'. The error of points was less than the set tolerance of 0.05. Thus, the enhancement to the algorithm provided by the integration of agent-based concepts is shown to yield better performance (i.e., less function evaluation needed to find the optimal points within a predefined interval). Additionally, several other functions more closely related to the real-world scenarios i.e., load profile functions representing the peak load demand at different times of a day were considered. As an example, based on the results obtained via the algorithm at work, during the execution of the utility load profile function represented by $y_1 = 0.941\ x^6 - 7.3795\ x^5 + 21.2895\ x^4 - 28.0191\ x^3 + 16.540\ x^2 - 2.8859\ x + 0.6294$ with time interval 0 to 2.4 (x10 hrs), step size of 0.005 and tolerance of 0.05, the results indicated better performance of the 'Reinforcement Learning and Agent-based Search' versus 'Reinforcement Only' for the utility function y_1 being evaluated. The improvement factor was 9.47 for the number of iterations (i.e., 13.3 versus 126 iterations) and 3.75 for the number of function evaluations (i.e. 68.1 versus 256). Furthermore, the maximum value P_{max} with respect to time of a day (hrs) and energy usage (%) was (18.5, 167.74) and the error of maximum value found via algorithm P_{max} (18.3, 167.67) was much less than the set Tolerance (i.e., time of a day: 0.05 x 10 = 0.5 hrs, energy usage: 0.05 x 100 = 5 %). Similar results were obtained for other utility load profile functions, reflecting different patterns of possible peak load demand.

The 'House Simulator' (Chapter 5) was developed as a tool to simulate house heating/cooling systems (equipped with PCTs) under different scenarios and for different TOU rates applied. More importantly, the role of a 'House Simulator' was to assist in implementation and evaluation of the proposed OLA algorithm; thus, representing an 'expert system shell' whereas the main blocks consist of a 'House Simulator,' a 'Smart Thermostat,'

and a placeholder for 'Adaptive Learning' models to be implemented and evaluated. The approximate energy savings of the scheduled set point settings (refer to Tables 2 and 3) versus fixed set point (20 ºC) for a period of three months demonstrated potential energy savings of 906 KWh. Moreover, the comparison of simulation results for the zone controlled house (i.e., scheduled set points for zone 1, and 17 ºC for zone 2) versus the entire house (with scheduled set points), confirmed additional energy savings of 614 KWh.

Furthermore, a novel adaptive learning algorithm 'Observe, Learn and Adapt' based on the ALS model was proposed and implemented (Chapter 6). Its evaluation was achieved with the aid of a tool being developed for this purpose i.e., 'House Simulator' (Chapter 5). Based on the ASHRAE recommendations [AME09], the thermal model for a house was derived and 'Simulation Engine' implemented. The Knowledge-Base technique for Zone Controlled Environment (i.e., airflow rate and heater adjustment) was proposed. It was demonstrated via emulation of real-world scenarios, that the OLA algorithm is a practical implementation reflecting the main features of the ALS learning technique. In configurations, such as of the entire house with OLA enabled versus entire house with OLA disabled, for the duration of 30 days and minimal changes, the results of OLA's performance assessment, showed potential energy savings of 79 KWh. While in comparison to configurations which consider a zone controlled OLA enabled house, provides an additional 279 KWh of energy savings. Thus, the improvements with respect to relative energy consumption/savings with and without application of OLA algorithm are obvious. It has to be noted that the Knowledge-Base implementation for the zone controlled environment (depicted in a general form in 1.17), is an integral part of the overall OLA's design.

Moreover, in order to further evaluate and validate the performance of OLA, a 30 day simulation was executed with FEL active and inactive (i.e., with and without input from sensors) where several occupants' pattern/schedule changes occurred. In the case of OLA with FEL inactive (i.e. no sensors, incapable to observe and adapt) the system is unable to detect the occupant's changing patterns/schedules, while in the case of OLA with FEL active, this is possible, and therefore is utilized to adapt and apply the new pattern changes to the current schedule. For a small change of only two Set Point end times for daily clusters of Monday: Set Point 2 changed from 6:00 to 8:00 AM, to 6 to 7:30 AM, respectively for Tuesday Cluster from 6:00 to 8:00 AM, to 6:00 to 7:15 AM, for only two occurrences of the

above scenario within a 30 day period (since the learning rate of OLA is 3 weeks before adapting to the changes) the estimated savings for a daily adapted pattern was 5 KWh for an OLA enabled house with zone controlled and FEL active versus configuration with FEL inactive; which in fact is 152 KWh when compared to a configuration without zone control, with FEL inactive and OLA disabled. The resulting changes were observed in the Master Cluster file, and scheduled event occurrences, which took place after the changes were adapted, validating the successful adaptation of OLA to the changes. Further experiments were conducted with a multitude of actual Heat Set Points and start and end time changes, and the adapted values taking effect were obvious (as described in detail in Chapter 6, Section 6.5).

The results of OLA's performance evaluation confirm improvements of an actual OLA enabled house with respect to energy savings, and validate the ALS model implementation via OLA. In addition, it also brings to fruition the concept of a 'Smart Thermostat' - a PCT with enhanced learning and adapting capabilities (based on the occupant's input pattern changes and other preferences). Moreover, a zone controlled OLA enabled house with FEL active (i.e., equipped with sensors), shows further performance improvements with respect to optimized energy conservation and comfort.

7.2 Future Research

The convergence of AI techniques and WSN, and their blend into AmI (perhaps the next step in AI's evolution) indicate an interesting area to be considered and explored further for numerous applications in Smart Homes and Buildings.

Moreover, finding an intelligent solution for optimal tuning of the zone controlled heating/cooling system parameters, which complements the OLA and Knowledge-Base of a 'Smart Thermostat' presented in this thesis, could be very practical and challenging interdisciplinary research (i.e., taking into account that in addition to size, type of materials used, layout, orientation, number of occupants, type of heating/cooling system used, thermal dynamics of each building are different). Possible further improvement of OLA to be considered might be the addition of a fuzzy logic approach for potential self-tuning of offsets related to control of airflow rate and heating/cooling stages applied for different zones i.e., within large buildings. Investigation of fuzzy logic in conjunction with rules-based technique

application could prove to be an effective approach to be considered in this case. On the other hand, consideration of OLA's behavior during seasonal transitions could be another interesting issue to be considered for further improvement. Although, typical schedules for the first occurrence of each season can be chosen as a starting point for the learning process, other approaches might be worth while investigating.

The concept of a 'Smart Grid' which extends beyond the work presented in this thesis is an extension which could be a viable direction to follow in order to achieve interoperability of different systems on a bigger scale. The ALS model can be utilized to further extend the applicable research of 'Adaptable Systems for Smart Buildings' into 'Smart Grids'. One key point to be mentioned is further exploration of AmI and non-intrusive means to incorporate autonomous distributed sensors i.e. 'intelligent agents' within Smart Homes and Buildings, which in addition to complementing the 'Smart Thermostats', assist in the overall picture of 'Smart Grid' by helping in peak load adjustment (i.e. controlling of smart appliances, load control switches, plug-in hybrid electric vehicles, lighting control, HVAC systems, etc.) during peak load hours and/or other times. Managing and/or predicting the peak load scenarios, while acting in a harmonized manner within different geographic areas of a city and/or state, where demands are not necessarily uniformly distributed, could pose an interesting challenge to address; it could encompass EGU, SCADA systems, Smart Meters, 'Smart Thermostats', 'intelligent agents' and/or other. In addition, the application of RL and Agent-based technique such as presented in Chapter 4 could be further extended by adding more 'awareness' and intelligence to sensor/actuator nodes (related to control of appliances, load control switches, etc), which could aid in load curtailment. In addition, the algorithm might be further enhanced by considering its implementation and feasibility in three dimensional spaces.

BIBLIOGRAPHY

[ALT85] J.L. Alty, *"Use of Expert Systems,"* IEEE Journal of Computer-Aided Engineering, Vol. 2, No. 1, pp. 2-9, Feb. 1985.

[AME09] American Society of Heating, Refrigerating and Air-Conditioning Engineers, *"ASHRAE Handbook - Fundamentals,"* (I-P Edition), USA, 2009, ISBN 978-1-933742-54-0.

[AMI10] G.P. Amis and G.A. Carpenter, *"Self-supervised ARTMAP,"* Elsevier Journal of Neural Networks, Vol. 23, No. 2, pp. 265-282, 2010.

[AMI05] F. Amigoni, N. Gatti, C. Piciroli and M. Roveri, *"What Planner for Ambient Intelligence Applications?,"* IEEE Transactions on Systems, Man and Cybernitics – Part A: Systems and Humans, Vol. 35, No. 1, pp. 7-21, Jan. 2005.

[ARA00] S. Arai, K. Sycara and T.R. Payne, *"Multi-agent Reinforcement Learning for Planning and Scheduling Multiple Goals,"* in Proc. of IEEE 4th International Conf. on Autonomous Agents, pp. 359-360, July 2000.

[ARE05] Arens, E., C.C. Federspiel, D. Wang, and C. Huizenga, 2005. "How Ambient Intelligence Will Improve Habitability and Energy Efficiency in Buildings." *Ambient Intelligence,* eds. W. Weber, J.M Rabay and E. Aarts, Springer, March. pp.63-80.

[AUG07a] J.C. Augusto and D.G. Shapiro, *The first workshop on Artificial Intelligence Techniques for Ambient Intelligence* (AITAmI'06). AI Magazine, Vol. 28, No. 1, pp. 86-87, 2007.

[AUG07b] J.C. Augusto and P. McCullagh, *"Ambient Intelligence: Concepts and Applications,"* International Journal of Computer Science and Information Systems, Vol. 4, No. 1, pp. 1-28, 2007.

[BER91] H.R. Berenji, *"On the Integration of Reinforcement Learning and Approximate Reasoning for Control,"* in Proc. of IEEE 30th International Conf. on Decision and Control, Vol. 2, pp. 1900-1904, Dec. 1991.

[BER00] H.R. Berenji and D.A. Vengerov, *"Learning, Cooperation, and Coordination in Multi-Agent Systems,"* Technical Report, US-00-10, Oct. 2000.

[BI04] Z. Bi, T. Takashina, K. Tanaka and S. Watanabe, *"Exploring Agent Learning Process by using Mechanical Features in Agent-based Simulation,"* in Proc. of IEEE 2nd International Conf. on Intelligent Systems, Vol. 1, pp. 256-260, June 2004.

[BOG06] P. Bogg, *"Towards Information and Goal Based Agent Negotiation,"* in Proc. of IEEE International Conf. on Intelligent Agent Technology (IAT), pp. 613-617, Dec. 2006.

[BRA09] N.G. Brannon, J.E. Seiffertt, T.J. Draelos and D.C. Wunsch DC, *"Coordinated Machine Learning and Decision Support for Situation Awareness,"* Elsevier Journal of Neural Networks, Vol. 22, No. 3, pp. 316-325, 2009.

[BRD09] O. Brdiczka, J.L. Crowley and P. Reignier, *"Learning Situation Models in a Smart Home,"* IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, Vol. 39, No. 1, pp. 56-63, Feb. 2009.

[BYO08] K. Byoung-Kug, H. Sung-Kwa, J. Young-Sik and E. Doo-Seop, *"The Study of Applying Sensor Networks to a Smart Home,"* in Proc. of IEEE 4th International Conf. on Networked Computing and Advanced Information Management (NCM),pp. 676-681, Sept. 2008.

[CAR87a] G.A. Carpenter and S. Grossberg, *"A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition,"* in Journal of Computer Vision, Graphics and Image Processing, Vol. 37, No. 1, pp. 54-115, Dec. 1987.

[CAR87b] G.A. Carpenter and S. Grossberg, *"ART 2: Self-organization of Stable Category Recognition Codes for Analog Input Patterns,"* in Journal of Applied Optics, Vol. 26, No. 23, pp. 4919-4930, 1987.

[CAR91a] G.A. Carpenter, S. Grossberg and D. Rossen, *"Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System."* in Proc. of IJCNN-Seattle International Joint Conference of Neural Networks, Vol. 4, pp. 759-771, July 1991.

[CAR91b] G.A. Carpenter, S. Grossberg and J.H. Reynolds, *"ARTMAP: Supervised Real-time Learning and Classification of Nonstationary Data by a Self-organizing Neural Network,"* in Proc. of IEEE Conf. on Neural Networks for Ocean Engineering, Vol. 4, No. 5, pp. 565-588, Aug. 1991.

[CER02] A. Cerpa and D. Estrin, *"Ascent: Adaptive Self-configuring Sensor Networks Topologies,"* in Proc. of IEEE Infocom, pp. 1278-1287, June 2002.

[CHA08] M. Chana, D. Est'eve, C. Escriba and E. Campo, *"A Review of Smart Homes -Present State and Future Challenges,"* Elsevier Journal of Computer methods and programs in biomedicine, Vol. 91, No. 1, pp. 55-81, July 2008.

[CHE08] B. Chen and J. Wang, *"Design of a Multi-Modal and High Computation Power Wireless Sensor Node for Structural Health Monitoring,"* on Proc. of IEEE International Conf. on Mechtronic and Embedded Systems and Applications, pp. 420-425, Oct. 2008.

[CHO03] C.Y. Chong and S. P. Kumar*,"Sensor Networks: Evolution, Opportunities and Challenges,"* in Proc. of IEEE, Vol. 91, No. 8, pp. 1247-1256, Aug. 2003.

[COO09] D.J. Cook, J.C. Augusto and V.R. Jakkula, *"Ambient intelligence: Technologies, Applications, and Opportunities,"* Elsevier Journal of Pervasive and Mobile Computing, Vol. 5, No. 4, pp. 277-298, Apr. 2009.

[CUN05] R. Cunha, A. Silva, A. Loreiro and L. Ruiz, *"Simulating Large Wireless Sensor Networks Using Cellular Automata,"* in Proc. of the 38th Annual Simulation Symposium (ANSS), 2005.

[CUN98] A. Cunha and J. Neves, *"A Game-theoretic Approach to the Socialization of Utility-based Agents,"* in Proc. of IEEE International Conf. on Multi Agent Systems, pp. 413-414, July 1998.

[DOU09] J. Douglas, *"Thermal Comfort and use of Thermostats in Finnish Homes and Offices,"* Elsevier Journal of Building and Environment, Vol. 44, No. 6, pp. 1237-1245, June 2004.

[DOU94] J. Douglas, *"Smart Thermostats for Comfort and Conservation,"* EPRI Journal, Vol. 19, No. 2, pp. 20-23, Mar. 1994.

[DSO07] M. D'Souza, K. Bialkowski, A. Postula and M. Ros, *"A Wireless Sensor Node Architecture Using Remote Power Charging, for Interaction Applications,"* on Proc. of IEEE 10th Euromicro Conf. on Digital System Design Architectures, Methods and Tools, pp. 485-494, Aug. 2007.

[DUD79] R. Duda, J. Gaschnig and P. Hart, *"Model design in the PROSPECTOR Consultant System for Mineral Exploration,"* Expert Systems in the Microelectronic Age, Edinburgh Univ. Press, Edinburgh, Scotland, pp. 153-167, 1979.

[ELI07] J. Eliasson, P. Lindgren, J. Delsing, S.J. Thompson and Ch. Yi-Bing, *"A Power Management Architecture for Sensor Node,"* on Proc. of IEEE Conf. on Wireless Comm. And Networking (WCNC), pp. 3008-3013, March 2007.

[ELK08] A. El Kateeb, A. Ramesh and L. Azzawi, *"Wireless Sensor Nodes Processor Architecture and Design,"* on Proc. of IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), pp. 1031-1034, May 2008.

[FAU94] L. Fausett, *"Fundamentals of Neural Networks,"* Prentice Hall, USA, 1994, ISBN: 0-13-334186-0.

[FUN03] W. Fun and Y. Liu, *"Adaptive Categorization of ART Networks in Robot Behaviour Learning using Game-theoretic Formulation,"* Elsevier Journal of Neural Networks, Vol. 16, No. 10, pp. 1403-1420, March 2003.

[GEL01] E. Gelenbe, E. Seref and Z. Xu, *"Simulation with learning agents,"* in Proc. of the IEEE, Vol. 89, No. 2, pp. 148-157, Feb. 2001.

[GEO09] B. George, H. Zangl, T. Bretterklieber and G. Brasseur, *"Seat Occupancy Detection Based on Capacitive Sensing,"* IEEE Transaction on Instrumentation and Measurement, Vol. 58, No. 5, pp. 1463-1470, May 2009.

[GLO76a] S. Glossberg, *"Adaptive Pattern Classification and Universal Recoding I: Parallel Development and Coding of Neural Feature Detectors,"* Springer Journal of Biological Cybernetics, Vol. 23, No. 3, pp. 121-134, 1976.

[GLO76b] S. Grossberg, *"Adaptive pattern classification and universal recoding. II. Feedback, expectation, olfacation, and illusions*," Springer Journal of Biological Cybernetics, Vol. 23, pp. 187-202, 1976.

[GRA07] D. Graupe, *"Principles of Artificial Neural Networks,"* 2nd edition, World Scientific, Singapore, 2007, ISBN: 13-978-981-270-624-9.

[HAG08] H. Hagras, *"Employing Computational Intelligence to Generate More Intelligent and Energy Efficient Living Spaces,"* Springer International Journal of Automation and Computing, Vol. 5, No. 1, pp. 1-9, Jan. 2008.

[HAG04] H. Hagras, V. Callaghan, M. Colley, G. Clarke, A. Pounds-Cornish and H. Duman, *"Creating an Ambient-intelligence Environment using Embedded Agents,"* IEEE Journal of Intelligent Systems, Vol. 19, No. 6, pp. 12-20, 2004.

[HAR10] M. Harbers, J.J. Meyer and K. Van den Bosch, *"Explaining Simulations through Self Explaining Agents,"* Journal of Artificial Societies and Social Simulation, Vol. 13, No. 1, Jan. 2010.

[HAR07] S. Harte, B. O'Flynn, R.V. Martinez-Catala and E.M. Popovici, *"Design and Implementation of a Miniaturised, Low Power Wireless Sensor Node,"* in Proc. of IEEE 28th European Conf. on Circuit Theory and Design (ECCTD), pp. 894-897, Aug. 2007.

[HEA08] M. Healy, T. Newe and E. Lewis, *"Wireless Sensor Node Hardware: A Review,"* in Proc. of IEEE Sensors, pp. 621-624, Oct. 2008.

[HEI02] W. R. Heinzelman, A. P. Chandrakasan and H. Balakrishnan, *"An Application Specific Protocol Architecture for Wireless Microsensor Networks,"* IEEE Transactions on Wireless Communications, Vol. 1, No. 4, pp. 660-670, 2002.

[HES10] T. Hester, M. Quinlan and P. Stone, *"Generalized Model Learning for Reinforcement Learning on a Humanoid Robot,"* in Proc. of IEEE Conf. on Robotics and Automation (ICRA), pp. 2369-2374, May 2010.

[HIL00] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler and K. S. J. Pister, *"System Architecture Directions for Networked Sensors,"* in Proc. 9th International. Conf. on Architectural Support of Programming Languages and Operating Systems, pp. 93-104, 2000.

[HOL07] K. Hogler Karl and A. Willig, *"Protocols and Architectures for Wireless Sensor Networks,"* Wiley, England, 2007, ISBN: 978-0-470-51923-3.

[HON10] S. G. Hong, N.S. Kim, Ch. S. Pyo and W. W. Kim, *"Hybrid Sensor Module and Data Processing Using Low-Power Wakeup in WSN,"* on Proc. of IEEE 4th International Conference on Sensor Technologies and Applications (SENSORCOMM), pp. 191-195, July 2010.

[HOO10] K. Hoon and C. Ramos, *"A Survey of Context Classification for Intelligent Systems Research for Ambient Intelligence,"* in Proc. of IEEE International Conf. on Complex, Intelligent and Software Intensive Systems (CISIS), pp. 746-751, Feb. 2010.

[INF01] IST Advisory Group, *"Scenarios for Ambient Intelligence in 2010,"* European Commission Report, 2001.

[IZQ08] J. Izquierdo, I. Montalvo, R. Pérez and M. Tavera, *"Optimization in Water Systems: a PSO Approach."* Proceedings of the 2008 SpringSim Multiconference**,** Ottawa, Canada, 2008.

[JAV10] O. Javier and R. Lopez, *"Self-Organized and Evolvable Cognitive Architecture for Intelligent Agents and Multi-agent Systems,"* in Proc. of 2nd International Conf. on Computer Engineering and Applications (ICCEA), Vol. 1, pp. 417-421, March 2010.

[JIA08] H. Jian, D.Yumin, Z. Yound and H. Zhangqin, *"Creating an Ambient-Intelligence Environment using Multi-Agent System,"* in Proc. of International

Conf. on Embedded Software and Systems Symposia (ICESS), pp. 253-258, July 2008.

[JIN10] X. Jinlin; G. Qiang and J. Weiping, *"Reinforcement Learning for Engine Idle Speed Control,"* in Proc. of IEEE International Conf. on Measuring Technology and Mechatronics Automation (ICMTMA), Vol. 2, pp. 1008-1011, March 2010.

[JU04] J. Ju, M. Kamel and L. Chen, *"Reinforcement Learning and Aggregation,"* in Proc. of IEEE International Conf. on Systems, Man and Cybernetics, Vol. 2, pp. 1303- 1308, Oct. 2004.

[KAM94] E. Kam-Lum, *"Introducing Small, PC-based Expert Systems on a Limited Budget,"* in Proc. of IEEE International Electronics Manufacturing Technology Symposium (IEMT), Vol. 1, pp. 287-295, Sep 1994.

[KAN10] M. E. Kantarci and H.T. Mouftah, *"Using Wireless Sensor Networks for Energy-aware Homes in Smart Grids,"* in Proc. of IEEE Symposium on Comp. and Comm. (ISCC), pp. 456-458, June 2010.

[KOP07] A. M. Koplow and P. Wright, *"Design Architecture for Multi-zone HVAC Control Systems from Existing Single-zone Systems using Wireless Sensor Networks,"* in Proc. of SPIE, Vol. 6414, Jan. 2007.

[KUS07] N. Kushiro, T. Higuma and M. Nakata, K. Sato, *"Practical Solutions for Constructing Ubiquitous Network for Building and Home Control System,"* IEEE Transactions on Consumer Electronics, Vol. 53, No. 4, pp. 1387-1392, Nov. 2007.

[LI10] P.Li and M. Xianxi, *"An Improved ART1 Neural Network Algorithm for Character Recognition,"* in Proc. of IEEE Chinese Control and Decision Conference (CCDC), *2010*, pp. 2946-2949, May 2010.

[LIN02] C. Lin, C. C. Federspiel and D. M. Auslander, *"Multi-sensor Single-Actuator Control of HVAC Systems,"* in Proc. of Intl. Conf. for Enhanced Building Operations, 2002, pp. 14-18, Oct. 2002.

[LOC10] M.P. Locatelli, M. Loregian and G.Vizzari, *"Artificial Societies in a Community-based Approach to Ambient intelligence,"* Oxford Uni. Computer Journal, Vol. 53, No. 8, pp. 1152-1168, 2010.

[LYN05] C. Lynch and F. O'Reilly, *"PIC-based TinyOS Implementation,"* on Proc. of IEEE Second European Workshop on Wireless Sensor Networks, pp. 378-385, Jan. 2005.

[MAD02] S. Madden, M. J. Franklin, J. M. Hellerstein and W. Hong, *"A Tiny Aggregation Service for Ad-hoc Sensor Networks,"* in Proc. of 5th symposium on Operating Systems Design and Implementation (OSDI), Vol. 32, Spec. Issue, pp. 131-146, 2002.

[MEI08] A.K, Meier, *"Residential Thermostats: Comfort Controls in California Homes,"* eScholarship Repository, Lawrence Berkeley Nat. Lib., Univ. of California, Sep. 23, 2008 (http://repositories.cdlib.org/lbnl/LBNL-938E (Last accessed on Sep., 26, 2010).

[MIL06] J.C. Miles and A.J. Walker, *"The Potential Application of Artificial Intelligence in Transport,"* in Proc. of IEEE Intelligent Transport Systems, Vol. 153, No. 3, pp. 183-198, Sep. 2006.

[NAT08] Natural Resources Canada, HOT 2000 Simulator, CECT, (canmetenergy-canmetenergie.nrcan-rncan.gc.ca/eng/software_tools/hot2000.html), (Last accessed on Nov., 08, 2010).

[NEG05] M. Negnevitsky, *"Artificial Intelligence: A Guide to Intelligent Systems,"* Addison Wesley, England, 2005, ISBN: 0-321-20466-2.

[NEW07] R. Newman and J. Kemp, *"Developing Wireless Sensor Nodes for Real-World Applications,"* on Proc. of IEEE Conf. on Local Computer Networks, pp. 858-863, Oct. 2007.

[PAM10] C.R. Pamplona Filho, M.J. Cunha, F.M. Azevedo and G.L. Ferrari, *"Intellec System: Shell for Expert Systems Creation with Fuzzy Inference Machine Developed in Prolog,"* in Proc. of IEEE International Conf. of System Science and Engineering (ICSSE), pp. 521-524, July 2010.

[POS08] J.L. Posadas, J.L. Poza, J.E. Simo, G. Benet and F. Blanes, *"Agent-based Distributed Architecture for Mobile Robot Control,"* Journal of Eng, App. Of Artificial Intelligence, Vol. 21, No. 6, pp. 805-823, Sep. 2008.

[RAM08] C. Ramos, J.C. Augusto and D. Shapiro, *"Ambient Intelligence - the Next Step for Artificial Intelligence,"* in Proc. of IEEE Journal of Intelligent Systems, Vo. 23, No. 2, pp. 15-18, Apr. 2008.

[RAT07] C. Rattanaprateep and S. Chittayasothorn, *"A Frame-based Object Relational Expert Database System,"* in Proc. of IEEE International Conf. of AFRICON, pp. 1-7, Sep. 2007.

[RED06] R. Reddy, *"Robotics and Intelligent Systems in Support of Society,"* IEEE Journal of Intelligent Systems, Vol. 21, No. 3, pp. 24-31, May 2006.

[RED05] A. Redfern, M. Koplow and P. Wright, *"Design Architecture for Multi-zone HVAC Control Systems from existing Single-zone Systems using Wireless Sensor Networks,"* in Proc. of SPIE 13th International Conference on Intelligence Systems Application to Power Systems, Vol. 6414, 2005.

[REM05] P. Remagnino and G.L. Foresti, *"Ambient Intelligence: A New Multidisciplinary Paradigm,"* in Proc. of IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, Vol. 35, No. 1, pp. 1-6, Jan. 2005.

[REN03] Z. Ren and A.B. Williams, *"Lessons Learned in Single-agent and Multiagent Learning with Robot Foraging,"* in Proc. of IEEE International Conf. on Systems, Man and Cybernetics, Vol. 3, pp. 2757- 2762, Oct. 2003.

[RIC06] V. Ricquenbourg, D. Menga, D. Duran, B. Marhic, L. Delahoche and C. Logé, *"The Smart Home Concept: our Immediate Future,"* in Proc. 1st IEEE International Conference on E-Learning in Industrial Electronics, pp. 23-28, Dec. 2006.

[RUI09] Ch. Rui, H. Yi-bin, H. Zhang-qin and H. Jian, *"Modeling the Ambient Intelligence Application System: Concept, Software, Data, and Network,"*

IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, Vol. 39, No. 3, pp. 299-314, May 2009.

[RUS03] S. Russell and P. Norvig, *"Artificial Intelligence A Modern Approach,"* 2nd Edition, Prentice Hall, USA, 2003, ISBN: 0-13-790395-2.

[RUT05] U. Rutishauser, J. Joller and R. Douglas, *"Control and Learning of Ambience by an Intelligent Building,"* IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans, Vol. 35, No. 1, pp. 121-132, Jan. 2005.

[SAK07] S. Sakunia, S. Bhalerao, A. Chaudhary, M. Jyotishi, M. Dixit, R. Junade and R. PAtrikar, *"UbiSens: Achieving Low Power Wireless Sensor Nodes,"* in Proc. of IEEE International Conf. on Wireless and Opt. Comm. Networks (WOCN), pp. 1-6, July 2007.

[SAK93] S. Sakane, H. Okoshi, T. Sato and M. Kakikura, *"Distributed Sensing System with 3D Model-based Agents,"* in Proc. of IEEE International Conf. on Intelligent Robots and Systems Vol. 2, pp. 1157-1163, July 1993.

[SCH07] D.H. Scheidt and M.J. Pekala, *"Model-Based Agents,"* in Proc. of IEEE Power Engineering Society General Meeting, pp. 1-2, June 2007.

[SHE07] Y. Shen and B. Guo, *"Dynamic Power Management (DPM) based on Wavelet Neural Networks in Wireless Sensor Networks,"* in Proc. of IFIP International Conference on Network and Parallel Computing, 2007.

[SHE08] M. Shenassa and K. Khakpour, *"Knowledge Base Expert System For Tuning PID Controllers Using Wireless Technology,"* in Proc. of IEEE International Conference on Comp. and Comm. Eng.(ICCCE), pp. 310-313, May 2008.

[SHO76] E. H. Shortliffe, *"Computer-Based Medical Consultations: MYCIN,"* Elsevier Press, New York, 1976.

[SIL08] J. Silva, M. Lima, G. Campos and J. Souza, *"A Fuzzy Utility-based Agent for the Design Problem,"* in Proc. of IEEE International Conf. of Engineering Management (IEMC), pp. 1-5, June 2008.

[SMI94] S.D.G. Smith and R.A. Escobedo, *"Engineering and Manufacturing Applications of ART-1 Neural Networks,"* in Proc. of IEEE International Conf. on Neural Networks, World Congress on Computational Intelligence, Vol. 6, pp. 3780-3785 Jun 1994.

[SON98] K.T. Song and T. S. Chu, *"Reinforcement Learning and its Application to Force Control of an Industrial Robot,"* Control Engineering Practice, 1998, Vol. 6, No. 1, pp. 37-44, 1998.

[SUH08] C. Suh and Y. B. Ko, *"Design and Implementation of Intelligent Home Control Systems based on Active Sensor Networks,"* IEEE Transactions on Consumer Electronics, Vol. 54, No. 3, pp. 1177-1184, Aug. 2008.

[TAN09] Y. Tan, W. Liu and Q. Qiu, *"Adaptive Power Management using Reinforcement Learning,"* IEEE/ACM International Conference on Computer-Aided Design Digest of Technical Papers, 2009.

[TUD10] C. Tudor, A. Ionut, S. Ioan, D. Mihaela, C. Georgiana and M. Daniel, *"A Self-Adapting Algorithm for Context Aware Systems,"* in Proc. of IEEE 9th International Roedunet Conference (RoEduNet, pp. 374-379, June 2010.

[VOJ08] A. Vojdani, "Smart integration," IEEE Power and Energy Magazine, Vol.6, No. 6, pp. 71-79, Nov. 2008.

[WAI01] G. Wainer and N. Giambiasi, *"Application of the Cell-DEVS Paradigm for Cell Spaces Modeling and Simulation,"* Journal of Simulation, Vol. 71, No. 1, pp. 22-39, Jan. 2001.

[WAI02] G. Wainer, *"CD++: a Toolkit to Define Discrete Event Models,"* Software, Practice and Experience, Wiley, Vol.32, No. 3, pp. 121-1306, Nov. 2002.

[WAN03] C.Y.Wan, S. B. Eisenman and A. T. Campbell, *"CODA: Congestion Detection and Avoidance in Sensor Networks,"* in Proc. of 1st ACM Conference on Embedded Networked Sensor Systems (SenSys), pp. 266-279, Nov. 2003.

[WAN10] C. Wang, S. Jin and H. Pang, *"Knowledge Processing of Instructional Intelligent Expert System,"* in Proc. of IEEE International Conf. on Optics, Photonics and Energy Engineering (OPEE), Vol. 1, pp. 215-217, May 2010.

[WEB05] W. Weber, J.M.Rabaey and E. Aarts, *"Ambient Intelligence,"* Springer, Netherlands, 2005, ISBN: 3-540-23867-0.

[WEN10] C. Wenbin, L. XiaoLing, L. YiJun and F. Yu, *"A Machine Learning Algorithm for Expert System based on MYCIN Model,"* in Proc. of IEEE 2nd International Conf. of Computer Engineering and Technology (ICCET), Vol. 2, pp. 262- 265, Apr. 2010.

[WIL06] E. Williams, S. Matthews, M. Breton & T. Brady, *"Use of a computer-based system to measure and manage energy consumption in the home,"* In Proc. of IEEE Intl. Symposium on Electronics and the Environment, pp. 167-172, May 2006.

[WIR00] S. Wiriyacoonkasem and A. Esterline, *"Adaptive Learning Expert System,"* in Proc. of IEEE SoutheastCon, pp. 445-448, Apr. 2000.

[WU08] X.Wu and T. Jiang, *"Matchmaking of Goals in Intelligent Agents based on Description Logics (DLs),"* in Proc. of IEEE International Conf. On Intelligent Computation Technology and Automation (ICICTA), Vol. 2, pp. 806-810, Oct. 2008.

[WU08] J. Wu and H. Qin, *"The Design of Wireless Intelligent Home System based on ZigBee,"* on Proc. of IEEE 11th International Conf. on Communication Technology (ICCT), pp. 73-76, Nov. 2008.

[YAM96] T. Yamaguchi, M. Masubuchi, K. Fujihara and M. Yachida, *"Realtime Reinforcement Learning for a Real Robot in the Real Environment,"* in Proc. of IEEE International Conf. on Intelligent Robots and Systems (RSJ), Vol. 3, pp. 1321-1328, Nov. 1996.

[YE02] F. Ye, G. Zhong, S. Lu and L. Zhang, *"Energy Efficient Robust Sensing Coverage in Large Sensor Networks,"* Technical Report, 2002.

[YEA07] E.M. Yeatman, *"Energy Scavenging for Wireless Sensor Nodes,"* on Proc. of IEEE 2nd International Workshop on Advances in Sensors and Interface (IWASI), pp. 1-4, June 2007.

[YEN02] G. Yen, T. Hickey, *"Reinforcement Learning Algorithms for Robotic Navigation in Dynamic Environments,"* in Proc. of IEEE International Joint Conf. on Neural Networks, Vol. 2, pp. 1444-1449, May 2002.

[YIG09] P. Yigong, L. Zhong-Cheng, Y. Jin-Shou, *"Multi-agent Framework for Energy Supply/Demand Prediction,"* in Proc. of WRI World Congress of Computer Science and Information Engineering, Vol. 4, pp. 586-590, March 2009.

[YUP07] T. Yu-Ping, H. Jun-Wei, L. Cheng-Ting and C. Chun-Yu, *"Building a Remote Supervisory Control Network System for Smart Home Applications,"* in Proc. of IEEE International Conference on Systems, Man, and Cybernetics, Vol. 3, pp. 1826-1832, 2007.

[ZEI02] B. Zeigler, T. Kim and H. Praehofer, *"Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems,"* Academic Press, 2000.

[ZHE06] Z. Zhenhuan and Y. Shuang-Hua, *"A Possible Hardware Architecture of Wireless Sensor Nodes,"* on Proc. of IEEE International Conf. on Systems, Man and Cybernetics, Vol. 4, pp. 3377-3381, Oct. 2006.

[ZHE07] Zh. Zhenhuan and Y. Shuang-Hua, *"Analysis of Power Consumption for Different Sensor Access Modes,"* on Proc. of IEEE International Conf. on Networking, Sensing and Control, pp. 329-333, April 2007.

APPENDIXES

APPENDIX A

SIMULATION OF LARGE WIRELESS SENSOR NETWORKS USING CELL-DEVS

A.1 Introduction

The advancement of electronic sensing devices, microcomputers and wireless communication devices has lead to creation of new smart sensors, which can monitor actuate, compute and communicate. Typically, these sensors are deployed in non-deterministic mode (randomly) when deployed in large numbers. These sensor devices have the capability to self-organize into the so-called Wireless Sensor Networks (WSN). WSN are ad-hoc networks, consisting of these spatially distributed sensing and processing devices. We introduce a model and a simulation study of Large Wireless Sensor Networks (WSN) by implementing the Topology Control Algorithm. We use the Cell-DEVS formalism, which enables efficient execution of cellular models. Thereafter, we observe and evaluate the behavior of sensor nodes and entire WSN from the simulation results obtained, under different test scenarios.

The emergence of powerful embedded micro-computer systems for wireless sensor networks provides a good ground for creation of new smart sensor systems, which can be useful to further promote new scientific endeavors and enhance our lives. Wireless sensors can monitor actuate, compute and communicate, yet are small in size and low in cost. The WSN are ad-hoc networks, consisting of these spatially distributed sensing and processing devices [HEA08]. WSN are used in many different applications, such as medicine, transportation and urban monitoring, traffic control, military, environment and habitat monitoring, energy management, smart homes, industrial applications, etc. The effectiveness of WSN is not just in their monitoring, actuating, computing and communications capabilities: with the added processing power, analog and digital ports, transceivers and memory, they can self-organize and communicate in the deployed area (as depicted in Figure A.1). Their processing power is limited however, and WSNs are usually deployed in large numbers and their load is shared accordingly.

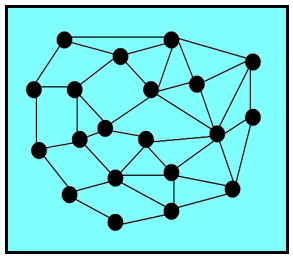


Figure A.1 - Sensor nodes self-organized

Due to the fact that sensor nodes have limited bandwidth, computing power and limited energy resources, one of the constraints in WSN is the energy efficiency of sensor nodes (i.e., their power consumption). There are many different approaches to solving the energy efficiency problem, such as the Topology Control Algorithm [CUN05] [YE02] [SHEN07]. The objective of this algorithm is related to the efficiency of WSN network, and it focused on how to increase its lifetime. The rationale of the network topology control algorithm is to reduce the number of redundant sensor nodes monitoring a particular region, hence increasing the efficiency and lifetime of the WSN network. In a nutshell, topology control exploits the redundant deployment of the sensor nodes, overcoming their energy limitations by restricting the set of nodes which are considered neighbors of a given node, while making sure that sensing area is still covered by a sufficient number of sensors. Furthermore, it reduces interference problems (which are noticeable when large number of sensor nodes are active).

We here introduce a model and a simulation of Wireless Sensor Networks (WSN) by implementing the Topology Control Algorithm using the Cell-DEVS formalism (which enabled efficient execution of this cellular model). The complexity of the problem can be simplified using the CD++ toolkit and the Cell-DEVS formalism (implementing this simulation in a other high level programming language is much more complex).

The rest of the work is organized as follows: we first give an overview of the Cell-DEVS formalism, followed by the WSN model definition. Thereafter, we observe and evaluate the behavior of sensor nodes and entire WSN from the simulation results obtained, under different test scenarios. We then describe and discuss the simulation results, analysis and discussion of the initial and improved results for different scenarios.

A.2 Model Definition

Due to the complexity of the model under study, we used Cell-DEVS [WAI02] and the CD++ toolkit [WAI02], as an efficient way to model and simulate cellular models [CHO98], in our case the WSN topology problem. Cell-DEVS is an extension to the DEVS formalism [8], which has been used to model systems that can be represented as cell spaces. A Cell-DEVS model is represented as a cell space, where each cell is represented as an atomic DEVS model. Each cell is connected to the local neighboring cells. A delay mechanism in each cell (transport delay or inertial delay) is used to delay the propagation of state change events through the cell space, providing the means for defining complex temporal behavior. An Atomic Cell-DEVS can be defined as follows:

$$TDC = < X, Y, I, S, \theta, N, d, \delta int, \delta ext, \tau, \lambda, D >$$

Where X is the set of external input events; Y is the set of external output events; I represents the definition of the model's modular interface; S is the set of possible states for a given cell; θ is the definition of the cell's state variables, N is the set of values for the input events; d is the delay of the cell; δint is the internal transition function; δext is the external transition function; τ is the local computing function; λ is the output function, and D is the duration function.

A Coupled Cell-DEVS model is built by connecting a number of Atomic Cell-DEVS models together into a cell space of any shape (including 2D and 3D cell spaces). The borders of the cell space can be either wrapped, in which case the cells at the border from one side of the cell space are considered neighbors to the cells at the border on the opposite side of the cell-space, or non-wrapped, in which case the border cells must have special rules defined by the modeler. A formal definition of Coupled Cell-DEVS is:

$$GCC = < Xlist, Ylist, I, X, Y, n, \{t1,\ldots,tn\}, N, C, B, Z, select >$$

Where Xlist is the input coupling list; Ylist is the output coupling list; I represents the definition of the model's modular interface; X is the set of external input events; Y is the set

of external output events; n is the dimension of the cell space; {t1,….,tn}is the number of cells in each of the dimensions; N is the neighborhood set; C is the cell space; B is the set of border cells; Z is the translation function; and select is the tie-breaking function;

CD++ is an M&S toolkit that implements DEVS and Cell-DEVS theory [WAI01] [WAI02][ZEI02]. Atomic models can be defined using a state-based approach (coded in C++ or an interpreted graphical notation), while coupled and Cell-DEVS models are defined using a built-in specification language. CD++ also includes an interpreter for Cell-DEVS models. The model specification includes the definition of the size and dimension of the cell space, the shape of the neighborhood and borders. The cell's local computing function is defined using a set of rules with the form: POSTCONDITION DELAY {PRECONDITION}. This indicates that when the *PRECONDITION* is satisfied, the state of the cell will change to the designated *POSTCONDITION*, who's computed value will be transmitted to other components after consuming the *DELAY*.

We have used these facilities to create an advanced model of WSN, in which we can analyze the WSN topology problem. This problem is closely related to the minimum configuration of nodes for fully operational WSN, taking into account sensor node energy limitations for a long lasting - survivable WSN networks. In our model, we have structured the area as a two-dimensional cell space of size ***n*** **x** ***n***, where every cell represents one sensor node. The WSN considered is such as all the sensor nodes have same properties (homogeneous WSN) and flat (i.e. no hierarchy among nodes). Each node can have up to 8 neighbors and there are ***3 possible states*** for each cell (*active*, *stand-by,* and *inactive*). For this model, Moore's neighborhood is adopted (i.e., the origin cell and its 8 close neighbors). As a result, the Cell-DEVS simulation model, gives an insight into an elegant way of implementation the Topology Control Algorithm for a large WSN; in our particular case, addressing the issue of sensor node energy conservation to achieve a longer lifetime operation of the WSN.

During the *active mode* of operation, the node emulates an active sensor within the WSN (i.e. performs processing tasks, hence using energy which decreases with time). In the beginning of simulation, all the sensor nodes deployed, have the same amount of energy which decreases as time progresses while node is in active mode (maximum energy consumption) or in stand-by mode (minimum energy consumption). During the *stand-by*

mode, sensor node is consuming a minimal amount of energy; it wakes up randomly in order to see if other close-by nodes are already sensing/monitoring the predefined neighborhood area. If less than two sensor nodes within the neighborhood are *active*, the cell goes again into *stand-by* mode; otherwise it becomes *active*. After the entire energy of a cell is consumed, the cell becomes *inactive*; this process continues until all the sensor nodes' energy is consumed (i.e. all the cells are inactive).

The ***n x n*** cells in the cell space considered were organized in two planes reflecting a three dimensional space implemented to meet the basic constraints (of the defined WSN Topology problem) while exploring and capturing the main tasks of the problem considered. Each sensor node starts to operate with a fixed amount of energy. In this model the energy levels adopted based on [CUN05] [YE02] are the following:

WSN sensor energy (at the beginning of operation) = 0.8 J

The energy of an WSN sensor node in ***active mode*** decreases by 0.0165 J every time step (in our case every 1 sec)

The energy of a WSN sensor node in ***stand-by mode*** decreases by 0.00006 J every time step (in our case every 1 sec)

The WSN sensor node possible states are:

Plane 0:

2 - WSN sensor node is in ***active mode*** within the neighborhood (WSN sensor coverage area)

1 - WSN sensor node is in ***stand-by mode*** within the neighborhood (WSN sensor coverage area)

0 – WSN sensor node is ***passive*** (i.e. energy of a node is consumed)

Plane 1:

0.8 - initial energy level of WSN sensor nodes

-1 - WSN sensor node is ***passive*** (i.e. energy of a node is consumed)

Neighborhood ={ (-1,-1,0) (-1,0,0), (-1,1,0), (0,-1,0), (0,0,0), (0,1,0)(1,-1,0),(1,0,0),(1,1,0), (0,0,1), (0,0,-1) }

Plane 0 contains the different deployed sensor nodes, and it is used to be observed throughout the simulation of the model. Plane 1 was used as "memory" for keeping track of the energy levels of active and stand-by nodes. The correct decrease of energy level and node behavior in the Plane 0 is interrelated to Plane 1 which contains the sensors energy information throughout the simulation.

The Plane 0 areas are organized as follows:

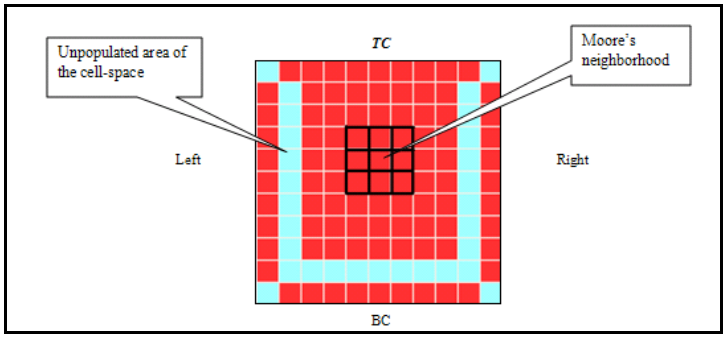


Figure A.2 - Plane 0 organization (zones)

TL - Top left rule, sensor node in the top left corner, i.e. origin coordinate (0, 0)

TC - Top center rule, sensor node coordinates (0, 1) to (0, 9)

BC - Bottom center rule, sensor node coordinates (10, 1) to (10, 9)

TR - Top right rule, sensor node coordinate (0, 10)

BL - Bottom left rule, sensor node coordinate (10, 0)

BR - Bottom right rule, sensor node coordinate (10, 10)

Right rule, sensor node coordinates (1, 10) to (9, 10)

Left rule, sensor node coordinates (1, 0) to (9, 0)

The rest of a cell space is WSN rule, i.e. local transition rule of the model.

The initial model was organized as shown in Figure A.2. The partitioning of the cell space into 'zones' was done to experiment with the problem at hand (initially) by isolating the 'zones'. Hence, easing the task to be solved, by observing sensors' operations at different 'zones' (and their interaction) prior to optimizing the final solution.

When compared to the initial model, the improved model (refer to Figure A.3) are the following:

Entire cell-space can be populated with sensor nodes

There are no limitations to functionality of the model (Plane 0 organization)

Cell space is not divided into zones

Enables more flexibility to the initial model (reduces significantly the code size)

Randomness was implemented within the model (it enables us to represent more closely the real-world applications related to WSN)

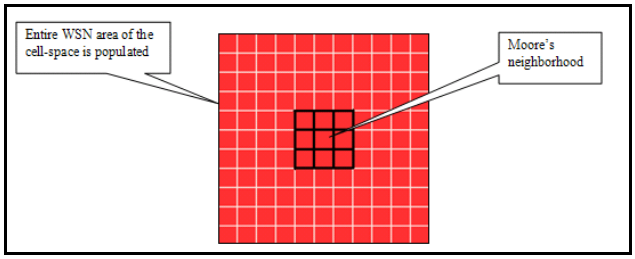


Figure A.3 - Plane 0 organization (improved model)

In addition to the existing states present in the initial WSN model (described earlier), an additional state was implemented in Plane 0 of the improved WSN model:

3 – WSN sensor node is typically in ***stand-by mode***; however, nodes randomly become active if one or two nodes are active

Within the neighborhood (Moore's neighborhood coverage area); node goes back to ***stand-by mode*** if the condition is true

(i.e. one or two neighborhood sensors are active), otherwise it remains in the ***active mode***.

A.3 WSN Behavior Definition

In Plane 0, each Moore's neighborhood (consisting of 9 cells) typically is covered by one active cell (with the value of 2) and the rest of the stand-by cells have a value of 1, whilst the passive cells have a value of 0. At the initial stage all the WSN sensor nodes deployed within the WSN network area (represented by cell space) are active for several time steps until they configure themselves to active and stand-by mode nodes.

In Plane 1, for every time step, the energy of active cells is decreased; the amount of energy available $\boldsymbol{x}$, for any active and stand-by mode cells before cell dies is $0 \leq \boldsymbol{x} \leq 0.8$. Each cells energy level in Plane 1 serves as a reference (memory) of the Plane 0 active and stand-by mode cells (represented by value of 2 and 1 respectively). Plane 0 refers to Plane 1 by (0, 0, 1), while Plane 1 refers to Plane 0 by (0, 0,-1) neighborhood coordinate (refer to

Figure A.4). The neighboring cells which are in a stand-by mode are represented by the value of 1 within the cell, and are also decreased in negligible quantities during the stand-by mode; The active cells become passive after their energy is consumed and are replaced by the neighboring cells which are yet alive (cells currently in stand-by mode).

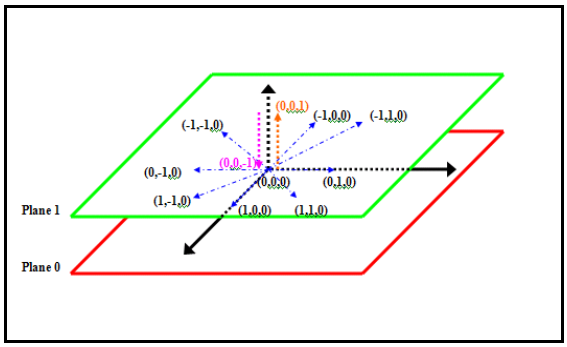


Figure A.4 - Cell space definition

The neighboring cells which are in a stand-by mode are represented by the value of 1 within the cell, and are also decreased in negligible quantities during the stand-by mode; The active cells become passive after their energy is consumed and are replaced by the neighboring cells which are yet alive (cells currently in stand-by mode). The cell-space in Plane 0 defines sensor nodes, whilst in Plane 1 energy level corresponding to each node (refer to Figure A.4).

The algorithm steps through each neighborhood and decides on which nodes stays active and which is to be configured as a stand-by nodes; where each neighborhood of cells is checked for present active cells by comparing the present cell's value with the residual energy (which is check referenced in Plane 1) until they becomes passive (represented by value of 0 in the cell space); Afterwards, one of the stand-by neighboring cell is assigned to becomes active, and this process continues until all the cells become passive (i.e. the energy resources of all sensors is consumed). Plane 1 updates the energy levels of each cell during the simulation as per the specifications.

The WSN simulation model provides closer approximation to the WSN topology algorithm by implementation of randomness within the WSN model; where randomness, redundancy and configuration of nodes play a significant role in reflecting the key factors in deployed networks, such as a network lifetime, coverage area and ratios of active and stand-

by sensors at specific points in time. Randomness was implemented by adding another rule (refer to the rule 3 below) to the model, where stand-by cells randomly become active if only one or two active neighbors are active. The actual results obtained with the improved model, more closely reflect the real-world scenarios and provide better insight into the WSN topology problem.

The problem could be reduced and coded with less than 30 lines of code utilizing the CD++ toolkit and Cell-DEVS, as shown in the following figure

```
[WSN]
type : cell
dim: (33, 33, 2)
delay : transport
border : nowrapped
neighbors :  (-1,-1,0)  (-1,0,0)  (-1,1,0) (0,-1,0)   (0,0,0)   (0,1,0)
neighbors :  (1,-1,0)   (1,0,0)   (1,1,0)  (0,0,1)    (0,0,-1)

localtransition : WSN-rule
[WSN-rule]
rule : { (0,0,0) - 0.0165 }  1000 {cellpos(2) = 1 and  (0,0,-1) = 2 }
rule : { (0,0,0) - 0.00006 } 1000 {cellpos(2) = 1 and (0,0,-1) =1  }
rule : { (0,0,0) - 0.0165 }  1000 {cellpos(2) = 1 and  (0,0,-1) = 3 }
rule : 3 1000 {cellpos(2)=0  and (0,0,1)>0 and
          ( (0,0,0)=1   and (statecount(2)=2 or statecount(2)=1) and randInt(30)=11 ) }
rule : 2 1000 {cellpos(2)=0  and (0,0,0) !=0 and (0,0,0) !=3 and (0,0,1) > 0 and (0,-1,0)!=2
         and (1,-1,0)!=2 and (1,0,0)!=2 and (1,1,0)!=2 }
rule : 1 1000 {cellpos(2)=0  and (0,0,0) !=0 and (0,0,1)>0 and
          ( (0,-1,0)=2  or (1,-1,0)=2 or (1,0,0)=2 or (0,1,0)=2 or (1,1,0)=2 or
          (cellpos(2)=0  and (0,0,0)=3) or (-1,-1,0)=2 or (-1,0,0)=2 or (-1,1,0)=2 ) }
rule : 0 1000 {cellpos(2)=0  and ((0,0,0)=1 or (0,0,0)=2 or (0,0,0)=0)  and (0,0,1) <= 0 }
```

A.4 Simulation Results

We executed numerous tests, and in this section we present some of the simulation results obtained and discuss their meaning. The Figure A.5 shows the graphical representation of the sensor node states defined in the simulation model:

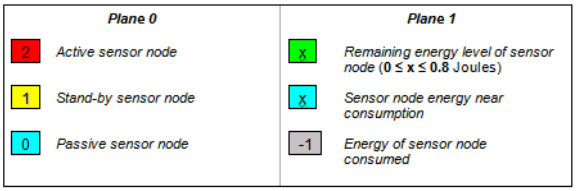


Figure A.5 - Possible sensor states of the initial WSN model

On the left side of the Figure A.5 we have the Plane 0 configuration, where **2** - indicates that a sensor node is ***active***, **1** – indicates that a sensor node is in ***stand-by mode*** and **0** - indicates that a sensor node is in ***passive state*** (i.e. energy of a node is consumed); while on the right side of the Fig.A5 we have the Plane 1 configuration, where **x** – is the energy level of WSN sensor nodes (green cells are the sensor nodes with enough energy, while blue color signifies that the sensor nodes are close to dying), **1** - WSN sensor node is ***passive*** (i.e. energy of a node is consumed)

The first simulation results presented are shown in Figure A.6 below. Based on the specification, we can see the sensor nodes active at the beginning of the simulation (red cells), while the light blue cells represent the unpopulated zones within the cell space.

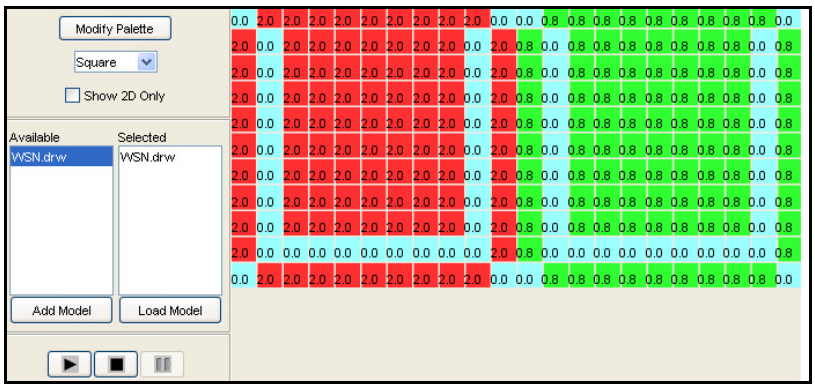


Figure A.6 - Plane 0 (left, red) and Plane 1 (right, green) prior to execution

The following Figure A.7 shows the WSN network during the process of reconfiguration, where nodes are trying to form a structure and every neighbor is trying to set some sensors in active mode while others remain in stand-by. The Plane 1 stores the energy level of each sensor node, which can be monitored.

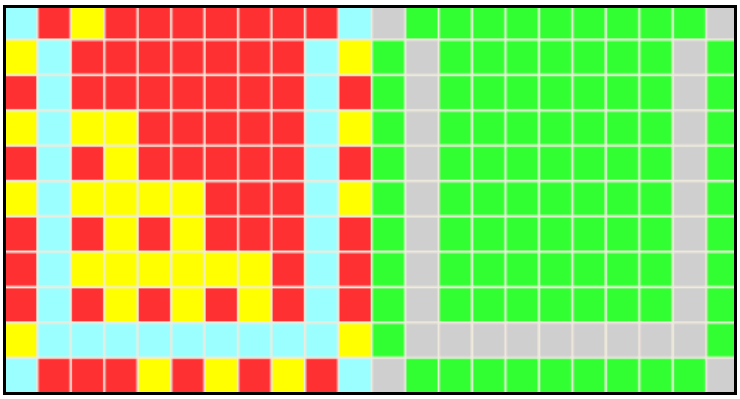


Figure A.7 - Snapshot of simulations results after 6 time steps

Figure A.8 shows that the WSN sensors are reconfigured as per the specifications and each neighborhood is covered by typically one active node while others remain in stand-by mode (energy of each note is decreasing during each step based on the model specification).

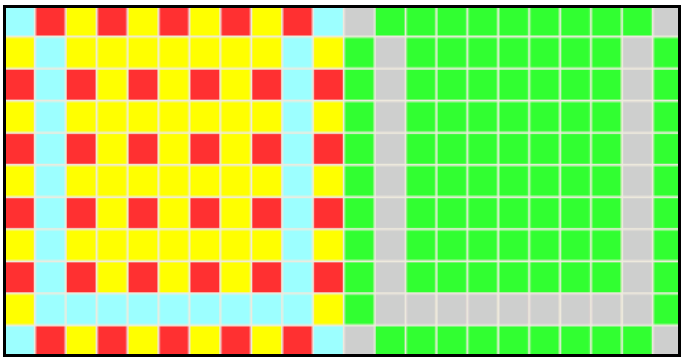


Figure A.8 - Snapshot of simulations results after 15 time units

Figure A.9 shows that several active nodes (from previous step) are now passive (died cells - Plane 1 on the right, gray cells), while cells that were previously in stand-by mode are taking over by becoming active.

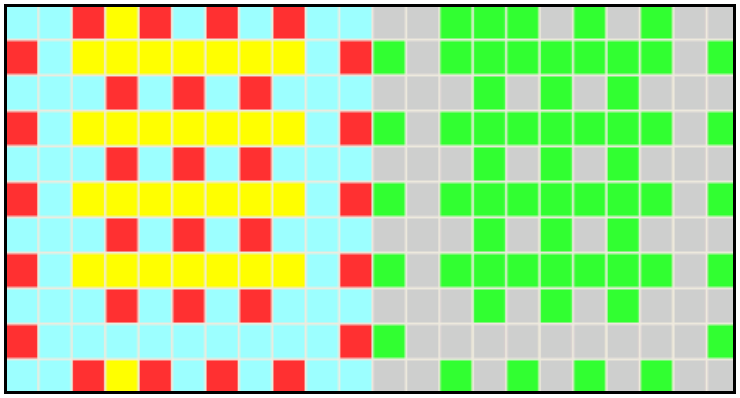


Figure A.9 - Snapshot of simulations results after 62 time units

In these simulations, all the sensor nodes become passive after 197 time steps. By comparing the end result of our simulation and the results obtained in [CUN05] (in particular, the number of active sensors after 200 time units), it can be observed that after approximately 200 time steps all the active sensors become passive (as energy of all nodes is consumed). Hence, the simulation results obtained by our model very closely reflect the same behavior (i.e. after 197 time units the active sensors become passive). These results can be seen in Figure A.10.

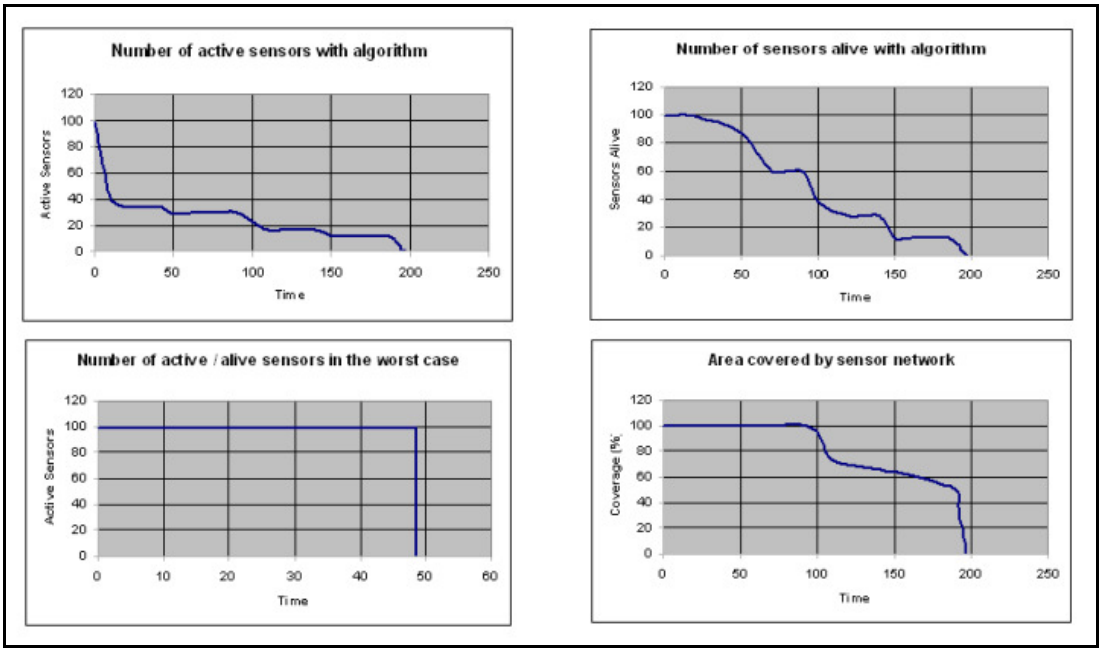


Figure A.10 - Initial WSN simulation results using Cell-DEVS

From the Figure A.10 it can be observed that in the worst case scenario (bottom left graph), when all nodes are active all the time, the sensor nodes die after 48.48 time units (where the energy of a node in each step is consumed by 0.0165 J and the initial level of sensors' energy is 0.8 J), while when using the Topology Control Algorithm, where selected sensor nodes within a Moore's neighborhood are alive (top left graph), the number of alive/active sensor nodes decreases gradually, extending the life of sensor nodes to 197 time units (top right graph) and the coverage area (bottom right graph).

Our following results present an improved version of the WSN model, in which we use the following is the graphical representation of the sensor node states defined within the simulation model:

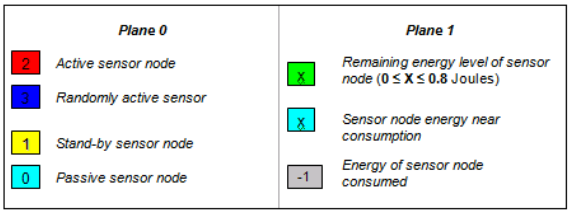


Figure A.11 - Possible sensor states of the improved WSN model

With respect to the initial model (refer to Figure A.5), in representing the sensor node states of the improved model, we have added an addition sensor state to indicate the randomly active sensor nodes, as depicted in Figure A.11.

This model provides closer approximation to the WSN topology algorithm by implementation of the randomness within the WSN deployed. The actual results obtained, more closely reflect the real-world scenarios and provide better insight into the WSN topology problem and how efficiently similar problems can be implemented utilizing Cell-DEVS and CD++ toolkit. The improved model is quite easy to modify in order to simulate different sizes of WSN networks. In this case, it requires to change only the values *n* and *m* (i.e., one line of specification, *dim : (n, m, 2)* where *n* is the number of columns and *m* is the number of rows in the model and provide the desired initial energy levels for the sensor nodes). The example below (refer to Figure A.12) represent the simulation model for WSN33, where the cell space is constituted by 33 rows and 33 column, and the total number of cells (sensor nodes) is 1089.

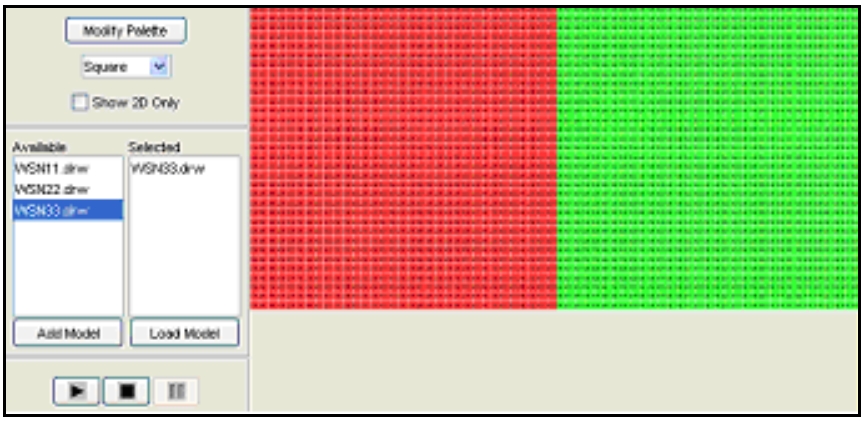


Figure A.12 - Snapshot of Plane 0 (left, red) and Plane 1 (right, green) prior to execution

The following Figure A.13 shows the 33 x 33 WSN network during the process of reconfiguration, where nodes are trying to form a structure and every neighbor is trying to set some sensors in active while others remain in stand-by mode. Some of the stand-by nodes are randomly awaken (in Fig.A13 left, the cells in blue in Plane 0). In case that only one or two sensor nodes are active, stand-by nodes randomly become active and return to stand-by mode only if one or two more sensor nodes are currently active within the neighborhood. On the Plane 1 data (refer to Figure A.13 right), we can see the energy level of each sensor node up to this time step (see in green, indicating that the energy levels x of sensor nodes are: $x \geq 0.05$J).

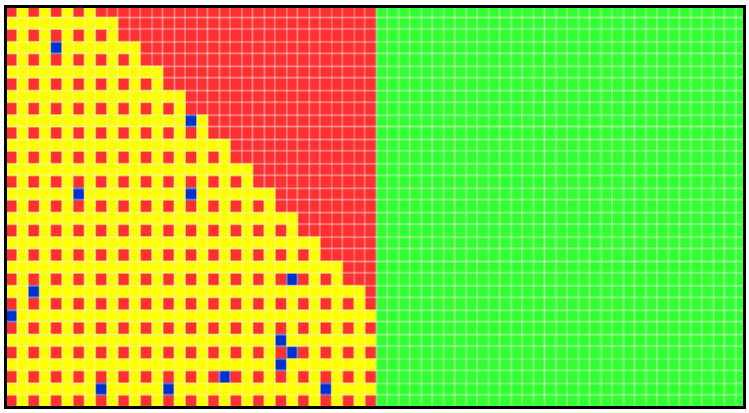


Figure A.13 - Snapshot of simulations results after 40 time units

Studying the simulation results of this Cell-DEVS model (presented in Figure A.14 and A.15), we can be see that the coverage area by sensors is reduced after 134 time steps, when more and more nodes become passive (as their energy is consumed). As time progresses, there is a smaller area of the WSN cell-space covered. Finally, after 193 time units, the WSN cell-space becomes passive. Similar, results were obtained when WSN22 and WSN11 were simulated.

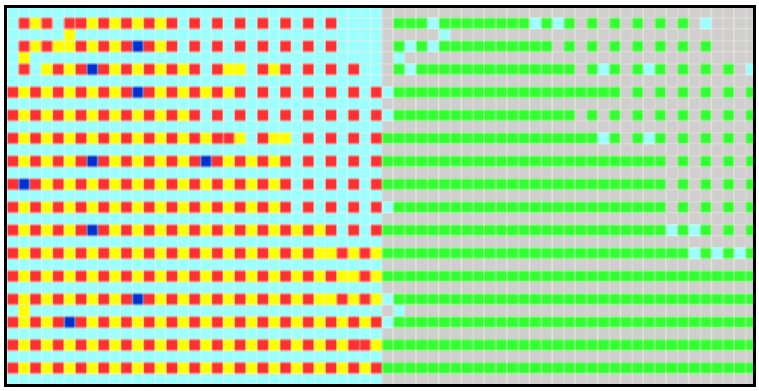


Figure A.14 - Snapshot of simulations results after 134 time units

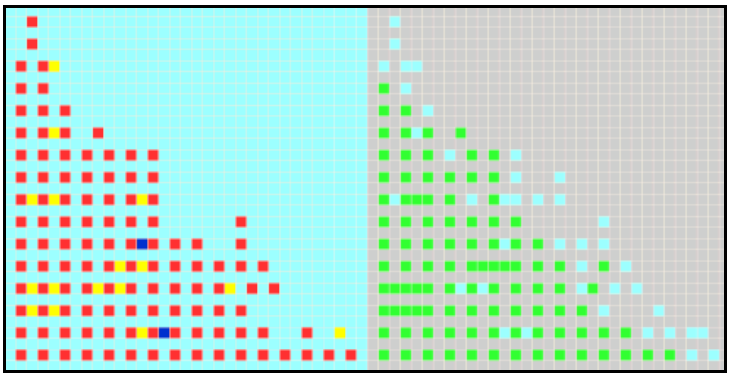


Figure A.15 - Snapshot of simulations results after 164 time units

Figure A.16 shows the number of active sensors versus time within the WSN network; evaluated for WSN deployment scenarios within two dimensional cell-spaces, using Cell-DEVS:

WSN11 - representing *11x11* cell-space, with 121 sensor nodes

WSN22 – representing *22x22* cell-space, with 484 sensor nodes

WSN33 – representing *33x33* cell-space, with 1089 sensor nodes

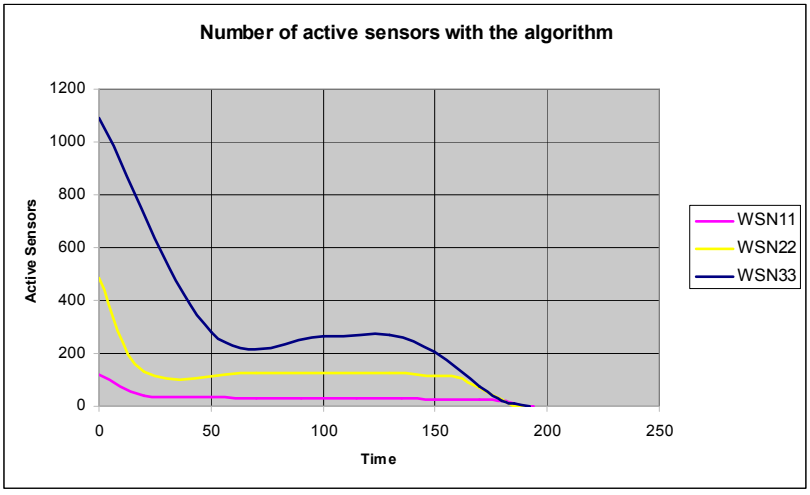


Figure A.16 - Number of active sensors with the algorithm using Cell-DEVS

As we can see, the number of active sensors decreases after the configuration of sensor nodes, following the deployment, after which, redundancy is reduced by having only one active node within the Moore's neighborhood (while other nodes are in stand-by). In addition, within each Moore's neighborhood, the stand-by nodes become active randomly and remain active if no sensor is active or return to stand-by mode if any sensor node is still active. The results obtained provide clear indication that network lifetime is increase approximately by 4, which was shown also in [CUN05].

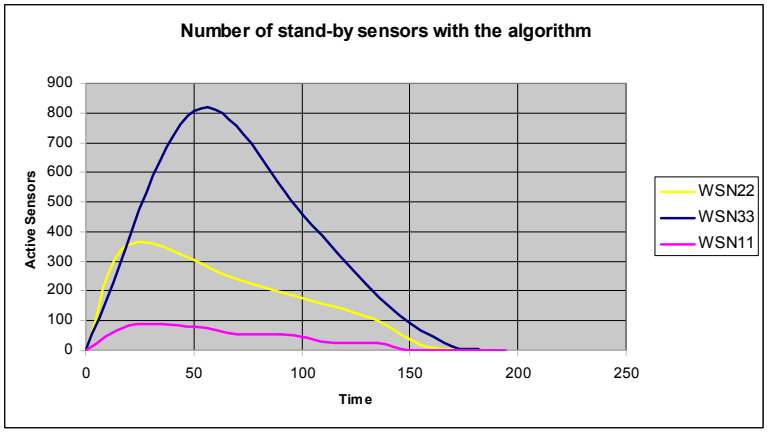


Figure A.17 - Number of stand-by sensors with the algorithm using Cell-DEVS

In Figure A.17, it can be observed that the number of stand-by sensors increases while the WSN cell-space is being configured, and starts to decrease as the active sensors' energy

is consumed (refer to Figure A.16) hence stand-by sensors become active. Similarly, in Figure A.18 we can be observe that the number of sensors alive is at it maximum when the simulation starts (i.e. all the WSN sensor nodes are active) and starts slowly decreasing as the time progresses.

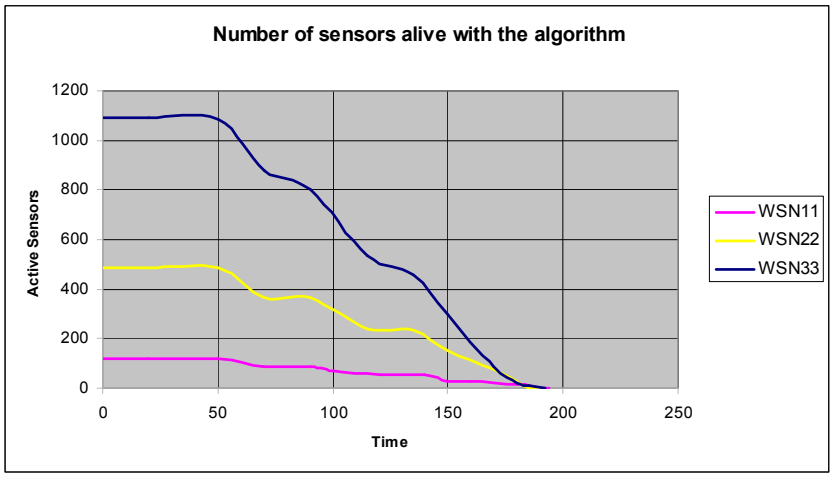


Figure A.18 - Number of sensors alive with the algorithm using Cell-DEVS

An important aspect considered in WSN networks is the coverage area, which is closely related to the active sensors within the cell-space. When the redundancy of sensor nodes within the WSN is controlled, the network lifetime is prolonged; hence wider coverage area is maintained for a longer time, as shown by the simulation results in Figure A.19.

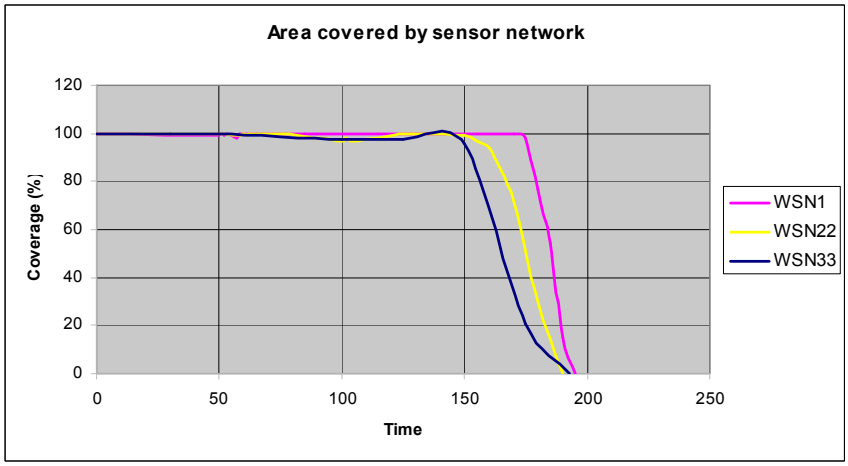


Figure A.19 - Sensor network coverage area (%) using Cell-DEVS

Finally, in Figure A.20 we can see the number of active sensors in the worst case scenario when WSN sensor nodes are active the entire time until the energy of sensors is consumed just before time 50; the network lifetime is approximately 4 time less in comparison to the implementation of WSN topology control algorithm using Cell-DEVS (refer to Figure A.16, A.18 and A.19).

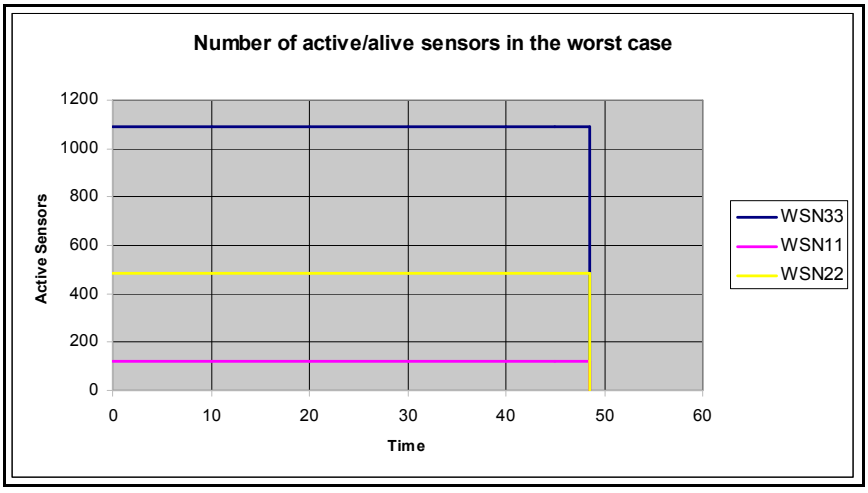


Figure A.20 - Number of active sensors with the algorithm

A.5 Conclusions

The objective of this research work was to simulate a large Wireless Sensor Network (WSN) using Cell-DEVS, by implementing the Topology Control Algorithm. By observing and evaluating the behavior of WSN simulation model, under different test scenarios, it was proven the effectiveness of Cell-DEVS and the CD++ toolkit, as an elegant approach to model, simulate and analyze, in this case the WSN topology problem.

The initial WSN model was further enhanced in order to provide closer approximation to the WSN topology algorithm by implementation of the randomness within the WSN deployed. In addition, modification and simplification of model was done, where cell-space is not divided into zones; allowing more flexibility to model any possible WSN cell-space configuration (entire or partial cell-space populated with sensor node).

The actual results obtained with the improved model, more closely reflect the real-world scenarios and provide a better insight into the WSN topology problem, in particular. As it was observed in section 3, the complexity of the problem can be simplified and coded with

less than 30 lines of code, utilizing CD++ toolkit and Cell-DEVS approach; whilst similar problem if implemented in C/C++ (or any other high level programming language) could possibly take up to several hundred or perhaps thousands of lines of code (based on the implementation approach taken). Thus, proving how efficiently complex problems of similar nature can be implemented simulated and analyzed utilizing CD++ toolkit and Cell-DEVS approach.

APPENDIX B

HEATER-COOLER SYSTEM PROTOTYPE

B.1 Introduction

Focus of this work is to design and emulate few potential features of a *Heater-Cooler System Prototype* by experiment (utilizing hardware and software).

In order to implement and experiment the main project ideas, in addition to the PICmicro®MCU 24FJ128GA006 which is a *Microchip's Technology* product, *Heater-Cooler System Prototype* utilizes the following peripheral devices:

One off board Temperature sensor (DS1631)

One on board Digital Potentiometer

Two on board Push Button(s)

Three onboard LED(s)

One off board 2x7 segment LED Display

Two off board LEDs

One off board Push Button

B.2 Brief Overview of PIC24F PICmicro® and Design Tools

B.2.1 PICmicro®MCU- PIC24F series

The PICmicro®MCU is *Microchip's* RISC based microcomputer that contains CPU, memory, oscillator and most of the peripherals inside a single integrated circuit. The PIC 24FJ128GA006 PICmicro®MCU adopted in the current design belongs to the Microchip's family of PIC24F series of general purpose microcontrollers based on FLASH technology.

The PIC24F series are 16-bit microcontrollers with modified Harvard architecture (with separate internal busses for memory and data). The enhancement of the PIC24F 16-bit CPU core with respect to the previous PIC18F family of the microcontrollers are the following:

- Uses 16-bit data bus and 24-bit address bus with the ability to move information from between data and memory spaces
- Linear addressing of up to 8MB for program space and 64Kbytes of data
- 16 working registers
- 17x17 hardware multiplier with support for integer math
- Hardware support for 32x16 bit division
- Barrel shifting/rotation up to 15 bits, left or right, shift or rotate
- Instruction set that supports multiple addressing modes and is optimized for high-level languages
- Operational performance up to 16 MIPS (million of Instructions per Second)
- Instruction clocking Tclk = Fosc/2 (previous families had Tclk = Fosc/4)

B.2.2 Design Tools

One of the most recommended high level languages for programming PICmicro®MCU is C, although other languages such as Pascal and PICBasic are commonly used. Many companies offer C compilers for PICmicro®MCU such as, IAR, Hi-Tech, CCS, etc.

In this work the CCS C Compiler for 24 bit PICmicro®MCUs. The CCS C compiler for 24-bit PICmicro®MCU family is called PCD compiler which supports dsPIC30, dsPIC33 and PIC24 family. The compiler implements efficiently typical C constructs, input/output operations and bit manipulations. All typical data types and many built-in (wrapper) functions are available for ease of use and are optimized for code efficiency.

The compiler's IDE is user friendly and comes with a build in debugger, providing good ground for debugging and code implementation. The In-Circuit Debugger (ICD-U40) on the other hand, enables programming and debugging or the PICmicro®MCU board (connects via USB to the PC and via ICD connector to the development board).

Main block diagram of the Design Tools used to accomplishment this project is depicted in Figure B.1.

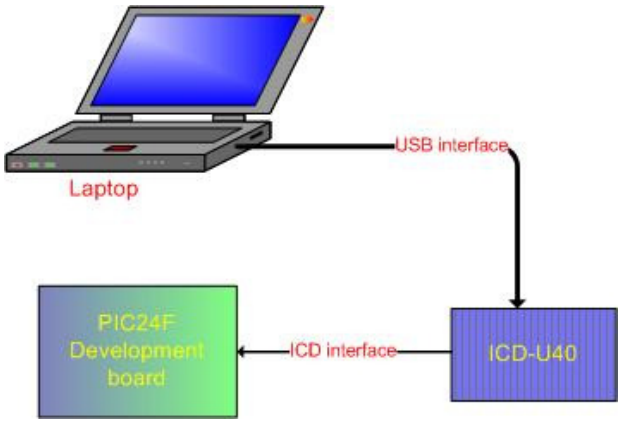


Figure B.1 - Block diagram of design tools used

B.3 Design Implementation of a System 'Emulator'

A simplified *Heater-Cooler Prototype System* was implemented in order to 'emulate' few of its potential features by experiment utilizing hardware and firmware. The proposed system was experimented utilizing PIC24F microcontroller development board suitable for this scheme, other additional components and integrated circuits. In addition to the hardware, for firmware development, Embedded C language for PICmicro® family from *Custom Computer Systems Inc.* (shortly CCS) was chosen.

Main system tasks covered in the experimental work done for the *Heater-Cooler System Prototype* would engage the following activities:

- Read the temperature sensor analog input (representing outside temperature) and display that information in LED segment display
- Utilize digital potentiometer to "emulate" inside temperature change (e.g. 0 to 5V representing different temperature levels in degree Celsius) hence, making system to decide and take actions accordingly
- Use LEDs to indicate different "system states" based on the level of temperature and data logging to monitor the actual temperature reading from sensor
- Use push button to indicate/emulate "room occupancy"; logic high indicating that someone is in the room hence, system acts by keeping the room temperature normal;

logic low indicating that room is empty, hence system drops the room temperature to minimum value

- Utilize interrupts within a system to maintain time, perform critical system tasks which do not fall into the category of sequential commands, hence require fast response (i.e. monitoring of buttons/switches pressed, de-bouncing, etc. during each interrupt e.g. every 1 millisecond)

B.3.1 Implementation of a Simple Test Bed for Heater-Cooler System Prototype

Figure B.2 below is a block diagram representation of a test bed for the *Heater-Cooler System Prototype*. Main components of the design are depicted in a simplified form.

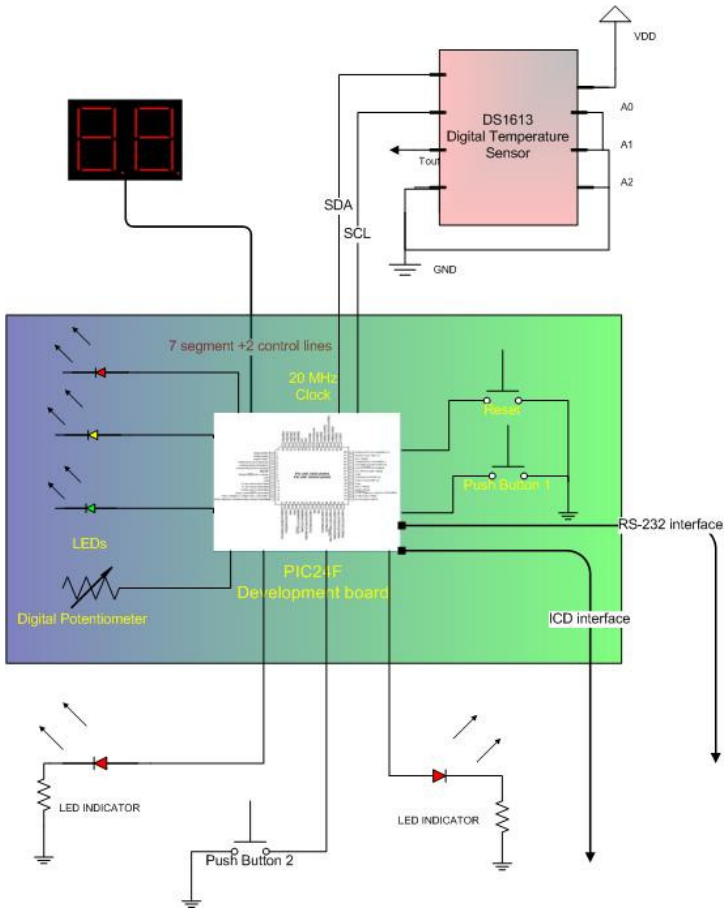


Figure B.2 - Block diagram of a system

24FJ128GA006 PICmicro® - is a 16-bit microcontroller offers good computational power and has a rich peripheral set of features desirable in the design of digital and embedded systems; hence can be utilized in the Heater Control System, as well.

The initial tasks included the selection of adequate input/output pins of the microcontroller, selection and configuration of the system clock to be used (setting of proper fuses); setting and configuration of the I/O port bits (direction), and also the analog to digital converter configuration.

The 20 MHz primary oscillator was chosen as a system clock.

Timer 1 Interrupts - The 24FJ128GA006 PICmicro® Timer 1 is a 16 bit register which was configured and used in the current design to monitor the critical tasks of the design. Taking into account the 20 MHz oscillator frequency and the execution cycle for typical instruction which for the PIC24F series which is 2 clock cycles proper 'system tick count' of 1 millisecond was implemented in a dedicated Interrupt Service Routine within the firmware (refer to the Appendix B, Timer1_ISR( ) source code).

1/20MHz/2 = 0.1 usec 'ticks' every instruction cycle; Since Timer 1 is a 16 bit register (0 to 65535), hence overflow occurs every 65535. In order to implement 1 msec interrupts Timer 1 register is loaded with the value of 65535, every time that a Timer 1 interrupt occurs.

Digital Temperature Sensor - The DS1631 digital temperature sensor was interfaced (via breadboard implementation) to the 24FJ128GA006 PICmicro® board for the emulated 'external temperature measurements'. The A to D conversion in this case was done by the DS1631, while the I^2C communication between DS1631 and the 24FJ128GA006 PICmicro® was used to communicate and receive the data. The DS1631 in this case was set as a slave and the 24FJ128GA006 PICmicro® was as a master. The processed data was used within the microcontroller to display the results and as a reference point for decision making process.

Digital Potentiometer - The 24FJ128GA006 PICmicro® analog port 0(sAN0) of the development board (having 10 bit resolution) connected to the digital potentiometer is chosen for the analog to digital conversion of the emulated 'internal temperature measurements'.

Push Buttons - There are 3 buttons within the system (two on board) and one implemented in the breadboard. The button 0 is the system reset button for the microcontroller. The on board Button 1 was implemented in the firmware (within the interrupt routine) to switch the displayed temperature from the external (reading from the DS1631 digital sensor in the

breadboard) to internal temperature reading (digital potentiometer in the development board) each time that Button 1 is pressed.

The Button 2 was implemented off board and interfaced to the PORTF pin F5 of the 24FJ128GA006 PICmicro®. It was set as input to sense when the Button 2 is pressed; which in turn toggles the off board RED LED 2 and RED LED 3 while the Button 2 is pressed. De-bouncing factor was considered and implemented not as a delay to the function but using the Timer 1 interrupts and msec counter in order to allow for several cycles of ON/OFF de-bouncing sequences prior to deciding that the button was pressed; De-bouncing period for the Button 2 (150msec) is different than the on board Button 1(100 msec); this was observed during the experimentation stage.

2x7 Segment LED Display – is implemented off board and interfaced to the PORTD pin D10 and D11 (for the control of segments to be lit) of the 24FJ128GA006 PICmicro®, and PORTB pins are utilized to energize individual LEDs within each segment respectively. Each corresponding LED segment a1 to a2, b1 to b2, c1 to c2, d1 to d2, e1 to e2, and f1 to f2 and g1 to g2 are connected together. Only seven wires connect each particular segment to the particular PORTB output port.

By controlling the common pins of the segment 1 and segment 2 (leaving a short delay (5 ms) in between ON and OFF commands issued to D10 and D11 – control lines of the segment one and two) using only 9 I/O lines, temperature values can be displayed.

LED Indicators – There are 3 on board LEDs (red, green and yellow) and two off board red LEDs. The on board LEDs are used to indicate different temperature levels (based on the reading from the emulated 'internal temperature', DS1631 temperature sensor reading. Such as, if the temperature reading is below 15 ºC Red LED indicator is activated (i.e. Heater ON); if the temperature reading is above or equal to 15 ºC and less than or equal to 25 ºC, yellow LED indicator is activated (i.e. Heater OFF and AC OFF). If the temperature reading is above 25 ºC green LED indicator is activated (i.e. AC ON).

The two off board red LEDs are implemented and interfaced to PORTE pins E0 and E1 respectively. Their function is to emulate extra health indicators and/or possible actuators related to the *Heater-Cooler System Prototype*. These LEDs toggle if the Button 2 is pressed. If the Button 2 is not pressed than the LEDs will lit if the emulated 'internal temperature' – i.e. analog to digital conversion value read from the digital potentiometer (as

we adjust the know). The settings of the LEDs is such as if the temperature reading is less than 25 ºC LED2 will lit; if the temperature reading is above or equal to 25 ºC LED3 will lit.

RS-232 communications – In addition to the ICD interface the development board has also RS232 interface. This interface was utilized for data logging during the system development stages. Section B.5 (Data Logging) shows the data from actual experimentation, where the 'Internal Temperature' is the temperature emulated by turning the digital potentiometer knob slowly from its minimum to its maximum position; while the 'External Temperature' is the temperature read from the DS1631 digital temperature sensor. Figure B.3 below, shows the actual experimental setup.

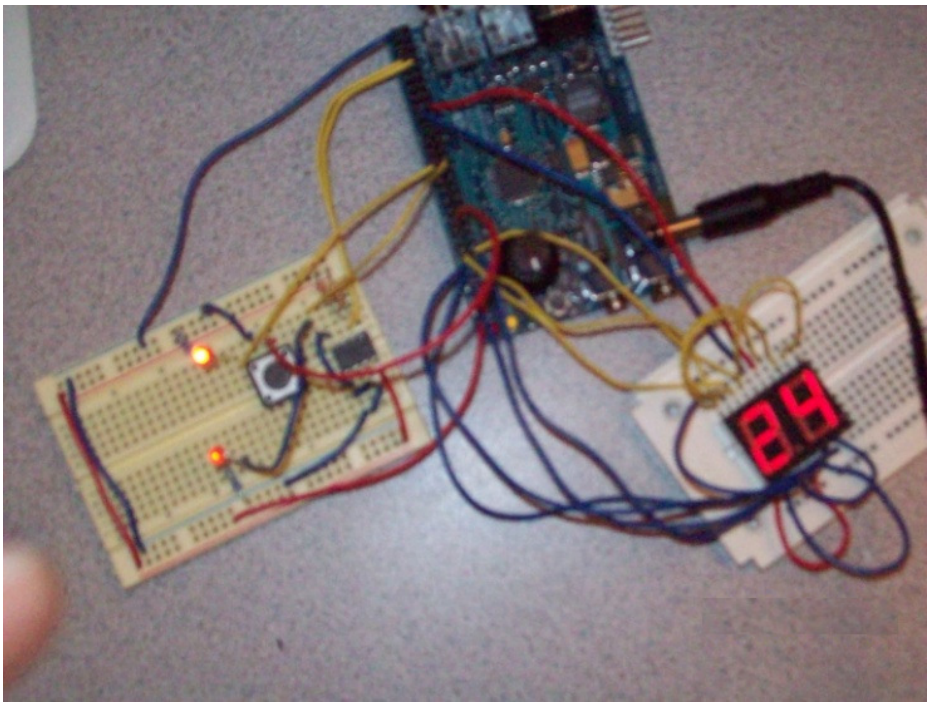
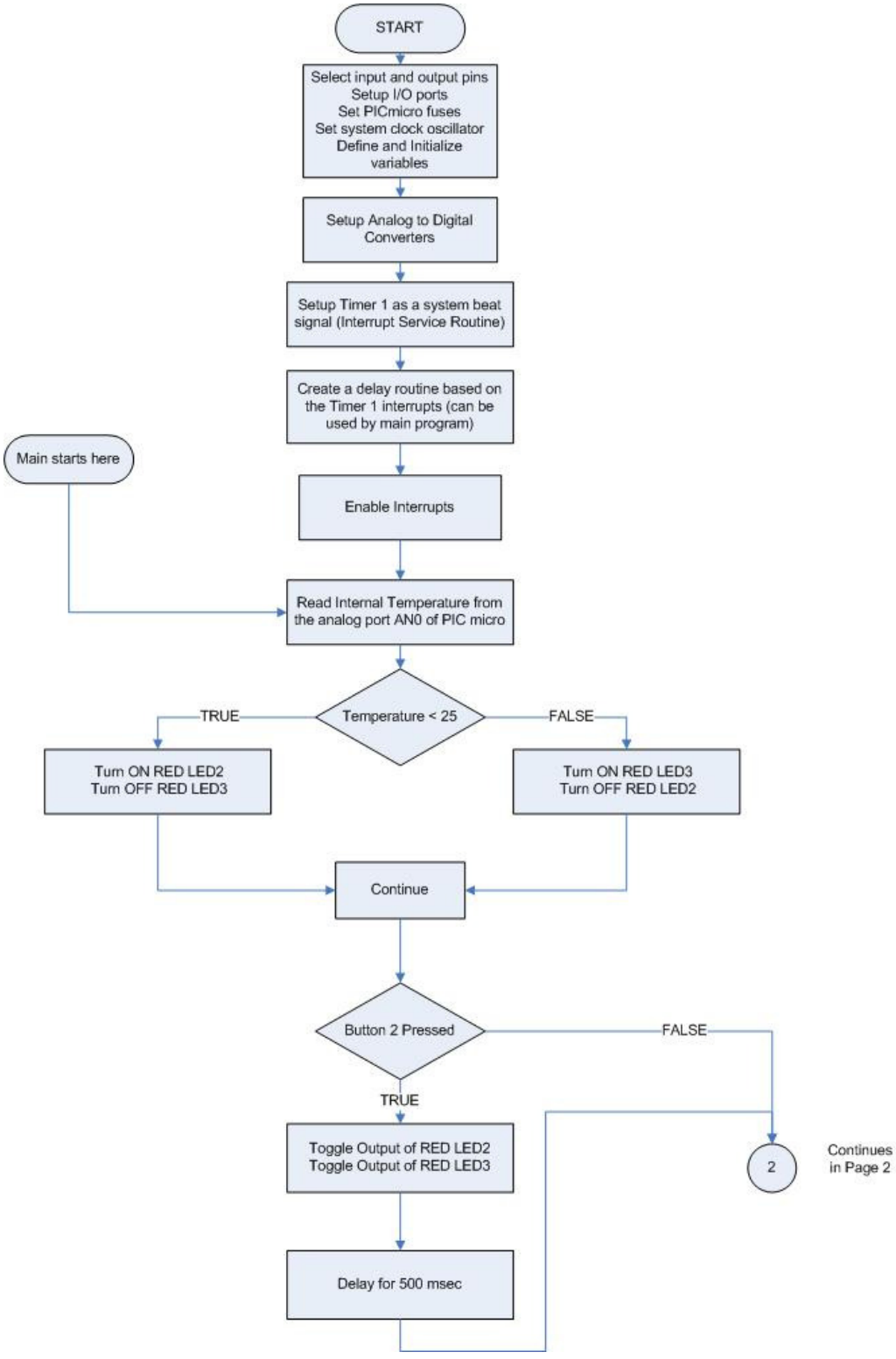


Figure B.3 – Experimental setup

B.4 Prototype Design Algorithm



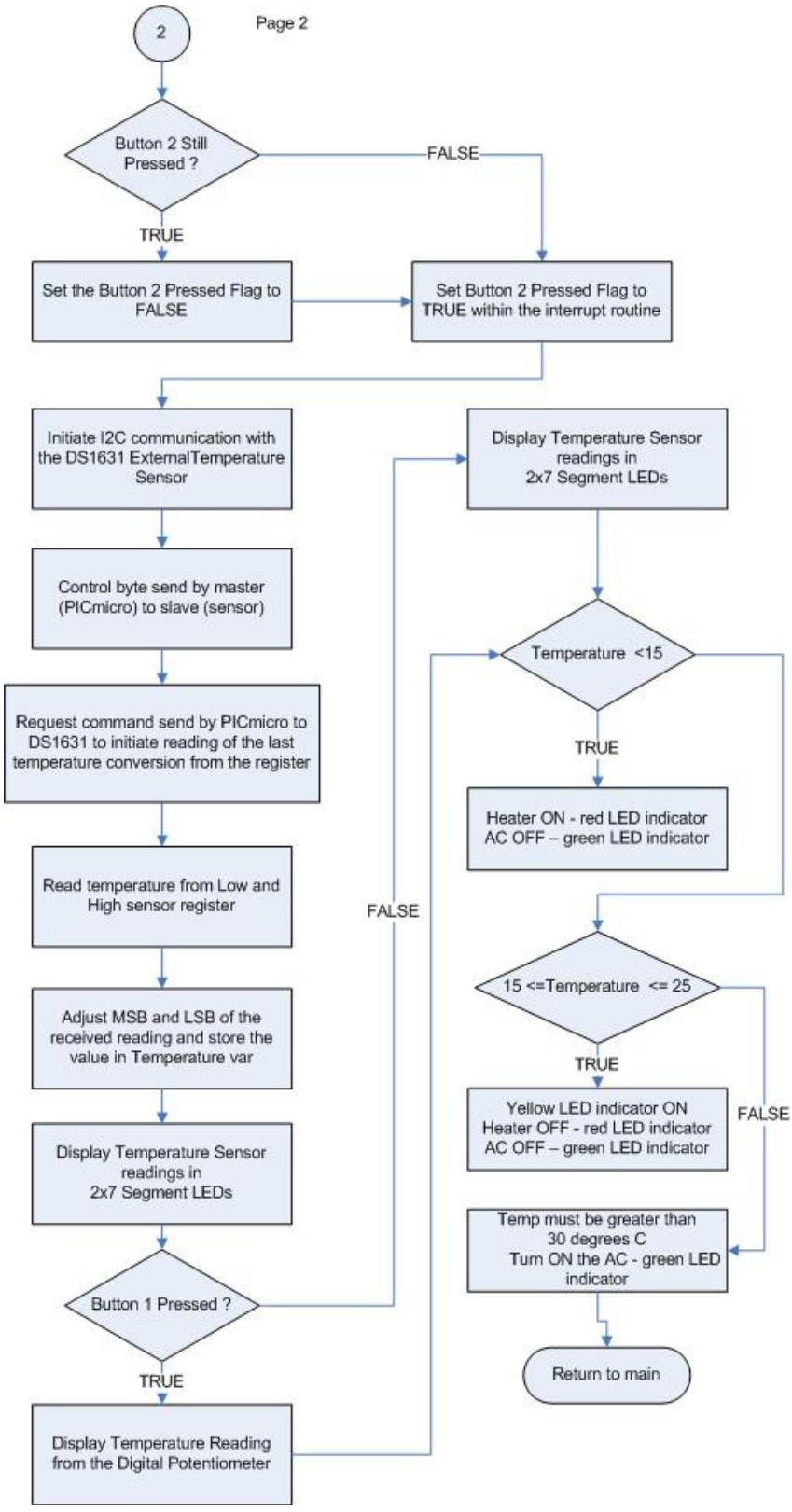


Figure B.4 - Prototype design algorithm

B.5 Source Code

```
/*
* File name              :  Heater_Sys.c
* Title                  :  Heater System
* Author                 :  Blerim Qela
*/
#include <24FJ128GA006.h>
#device ICD=TRUE
#device ADC=10
#fuses HS, NOWDT, PR
//#use delay (clock = 20000000)
//#use RS232 (baud=9600, UART1, stream=PORT1)
#define CLK PIN_D0
#define DATA PIN_D1
#use i2c(master, sda=DATA, scl=CLK)
#define GREEN_LED PIN_B5
#define RED_LED PIN_B2
#define YELLOW_LED PIN_B4
#define RED_LED2 PIN_E1
#define RED_LED3 PIN_E0
#define pushButton1 PIN_F6
#define pushButton2 PIN_F5
signed int16 readTemp;
signed int8 tempHigh, tempLow;
float Temperature, InternalTemp;
unsigned int16 digitalTemp;
unsigned int16 tick_count,msec;
unsigned int8 tp1,tp2, bDebounce;
int1 button1Pressed,button2Pressed;
```

```
//using timer 1 interrupt functionality
//create a delay routine (msec)
void my_delay(unsigned int16 x){
  unsigned int16 temp;
  temp = x;
  msec=0;

   do{
      #ASM NOP #ENDASM

   }while(msec<temp);

}
// array of 7 segment LEDs; each element activates a specific number
unsigned int16 const Segment_Leds[10]={0x3F00,0x0600,0x5B00,0x4F00,
0x6600, 0x6D00,0x7D00,0x0700,0x7F00,0x6700};
//Timer 1 interrupt routine
#int_TIMER1

void TIMER1_ISR()
{
  set_timer1(55535);          // overflow interrupt will occur approx. every 1msec
  tick_count++;               //increments tick counter
  msec++;                             // keep track of msec ticks

  //utilize timer to check for debouncing of button press
  if(!input_state(pushButton1)){
     bDebounce++;

     if(bDebounce >=100){
        button1Pressed = TRUE;
```

```
            bDebounce = 0;
        }
    }
    if(!input_state(pushButton2)){
        bDebounce++;

        if(bDebounce >=150){
            button2Pressed = TRUE;
            bDebounce = 0;
        }
    }
}

void main(){
    //setup timer 1 as system "tick" counter
    setup_TIMER1(TMR_INTERNAL|TMR_DIV_BY_1);

    //setup A to D converter for digital potentiomete
    setup_adc_ports(sAN0);        // setup port AN0 for reading the potentiometer
    setup_adc(ADC_CLOCK_INTERNAL); //The ADC will use internal clock
    set_adc_channel(0);
    set_tris_b(0x02);
    //initial setup sequence for reading temperature sensor
    output_high(DATA);       //bring SDA and SCL high
    output_high(CLK);
    i2c_start();           //initiate communication with sensor
    i2c_write(0x90);       //control byte send by master (micro)to slave (sensor)
    i2c_write(0x51);       //initiate one temperature conversion
    i2c_stop();            //stop command
      button1Pressed = FALSE;      // set button flag to 0 initially
    button2Pressed = FALSE;      // set button flag to 0 initially
```

```
bDebounce = 0;
msec=0;
  tick_count=0;
set_timer1(55535);          // overflow interrupt will occur approx. every 1msec
  enable_interrupts(INT_TIMER1);

while(TRUE){
  digitalTemp = read_adc(); //read analog port AN0 of PIC micro
  InternalTemp = (float)digitalTemp/10; // ADC value (0 to 1024)/10

  //check and turn on external LED indicators based on temp. settings

  if((digitalTemp < 25)&& !button2Pressed){

    output_high(RED_LED2);
    output_low(RED_LED3);

  }
  if((digitalTemp >= 25)&& !button2Pressed){

    output_high(RED_LED3);
    output_low(RED_LED2);
  }

  //toggle indicators/devices while button2 is pressed
  if(button2Pressed){
      output_toggle(RED_LED2);
      output_toggle(RED_LED3);
      my_delay(500);
      if(input_state(pushButton2)){
        button2Pressed=FALSE;
```

```
    }
 }
i2c_start();      //initiate communication with sensor
i2c_write(0x90);  //control byte send by master (micro)to slave (sensor)

//write command issued to DS1631 to read last temperature
//conversion from the register
i2c_write(0xaa);
i2c_start();        //initiating the read sequence
i2c_write(0x91);    //read command - requested by master
tempHigh=i2c_read(); //reading high byte of the temp register
tempLow=i2c_read(0); //reading low byte of the temp register
i2c_stop();

// adjust the MSByte of reading to high byte of the result
readTemp = (signed long)tempHigh*0x100;
readTemp = readTemp & 0xFF00;
readTemp = readTemp + tempLow;

//add the LSByte part
Temperature =(float) readTemp /256;
disable_interrupts(INT_TIMER1);
printf("External Temp:%3.2f\n\r",Temperature);
printf("Internal Temp:%3.2f\n\r",InternalTemp);
//delay_ms(1000);
enable_interrupts(INT_TIMER1);

tp1 = (unsigned int8)Temperature;
tp2= (unsigned int8)InternalTemp;

//display temperature in 7 segment LEDs
```

```
  //displays the most significant digit
  output_b(Segment_Leds[tp1/10]);
  output_high(PIN_D11);
  my_delay(5);
  output_low(PIN_D11);

  //displays the least significant digit
  output_b(Segment_Leds[tp1%10]);
  output_high(PIN_D10);
  my_delay(5);
  output_low(PIN_D10);

  tp2= (unsigned int8)InternalTemp;

  if(button1Pressed){
//press button1 to alternate 7 segment display to show actual digital potentiometer readings
      output_b(Segment_Leds[tp2/10]);
      output_high(PIN_D11);
      my_delay(5);
      output_low(PIN_D11);
      output_b(Segment_Leds[tp2%10]);
      output_high(PIN_D10);
      my_delay(5);
      output_low(PIN_D10);
            if(input_state(pushButton1)){
                  button1Pressed=FALSE;
          }
   } if (Temperature < 15){
    //Heater ON - red LED indicator
    output_high(RED_LED);
    output_low(GREEN_LED);
```

```
      output_low(YELLOW_LED);
      // my_delay(5);
    } else if((Temperature <= 25) && (Temperature >= 15)){
      //Heater OFF / AC OFF - yellow LED indicator
      output_high(YELLOW_LED);
      output_low(GREEN_LED);
      output_low(RED_LED);
      //my_delay(5);
    } else{
      // Temp must be greater than 30 degrees C
      // turn ON the AC - green LED indicator
      output_high(RED_LED);
      output_low(GREEN_LED);
      output_low(YELLOW_LED);
    }
  }

}
```

Data Logging File

04/25/2009 02:23:08.197 --> Internal Temp:0.00 //digital potentiometer – knob gradually turned
04/25/2009 02:23:09.239 --> External Temp:20.06 //reading from actual temperature sensor
04/25/2009 02:23:09.259 --> Internal Temp:1.00
04/25/2009 02:23:10.290 --> External Temp:20.00
04/25/2009 02:23:10.320 --> Internal Temp:1.10
04/25/2009 02:23:11.352 --> External Temp:20.00
04/25/2009 02:23:11.372 --> Internal Temp:1.79
04/25/2009 02:23:12.403 --> External Temp:20.00
04/25/2009 02:23:12.423 --> Internal Temp:2.29

04/25/2009 02:23:13.475 --> External Temp:20.06
04/25/2009 02:23:13.485 --> Internal Temp:2.50
04/25/2009 02:23:14.516 --> External Temp:20.06
04/25/2009 02:23:14.536 --> Internal Temp:2.89
04/25/2009 02:23:15.578 --> External Temp:20.00
04/25/2009 02:23:15.598 --> Internal Temp:3.20
04/25/2009 02:23:16.629 --> External Temp:20.00
04/25/2009 02:23:16.649 --> Internal Temp:3.79
04/25/2009 02:23:17.691 --> External Temp:20.00
04/25/2009 02:23:17.711 --> Internal Temp:5.00
04/25/2009 02:23:18.742 --> External Temp:20.00
04/25/2009 02:23:18.762 --> Internal Temp:6.50
04/25/2009 02:23:19.804 --> External Temp:20.00
04/25/2009 02:23:19.814 --> Internal Temp:8.80
04/25/2009 02:23:20.855 --> External Temp:20.00
04/25/2009 02:23:20.875 --> Internal Temp:21.50
04/25/2009 02:23:21.917 --> External Temp:20.00

References

1. N. Gardner, *"PICmicro MCU C – An Introduction to Programming the Microchip PIC in CCS C"*
2. T. Wilmshurst, *"Designing Embedded Systems with PIC Microcontrollers – Principles and Applications"*
3. CCS, Embedded C Language Development Kit for the PIC MCU by Custom Computer Systems, www.ccsinfo.com
4. Microchip's Technology, www.microchip.com
5. Dallas Semiconductor Maxim, www.maxim-ic.com

APPENDIX C

C.1 Master Cluster Structure

Master cluster structure is organized as follows:

1. Day of week/Cluster #: 1 for Monday, 2 for Tuesday, etc.
2. File number (1 to 10)
3. Currently active ? (1 = yes, 0 = no)Note: only one file can be active at a time
4. Nr. of days used (weight counter)
5. This file replaced file number # (0 means not replaced, e.g. 1 = replaced Monfile#1
6. SetPoint#1 (0 - not changed, 1 - changed)
7. SetPoint #2 (0 - not changed, 1 - changed)
8. SetPoint #3 (0 - not changed, 1 – changed)
9. SetPoint #4 (0 - not changed, 1 - changed)
10. SetPoint #5 (0 - not changed, 1 - changed)
11.Time when previous file was changed by this one (string - date and time) and 0 - means no time available

C.2 Daily Cluster Structure

Daily cluster structure is organized as follows:

1. 1 - Monday, 2 - Tuesday, etc.
2. SetPoint #
3. Heat SP
4. Cool SP
5. SP start time
6. SP end time
7. Zone1 Heat day Offset
8. Zone1 cool day offset
9. From (cZone1DayFrom.SelectedIndex)
10. To (cZone1DayTo.SelectedIndex)

11. Zone1 heat night offset
12. Zone1 cool night offset
13. From (cZone1NightFrom.SelectedIndex)
14 To (cZone1NightTo.SelectedIndex)
15. Zone1 Heat day Offset
16. Zone1 cool day offset
17. From (cZone1DayFrom.SelectedIndex)
18. To (cZone1DayTo.SelectedIndex)
19. Zone1 heat night offset
20. Zone1 cool night offset
21. From (cZone1NightFrom.SelectedIndex)
22. To (cZone1NightTo.SelectedIndex)
23. Maximum temperature offsets enabled by user during HIGH TOU rates - Heat Offset
24. Maximum temperature offsets enabled by user during HIGH TOU rates - Cool Offset
25. Maximum temperature offsets enabled by user during MEDIUM TOU rates - Heat Offset
26. Maximum temperature offsets enabled by user during MEDIUM TOU rates - Cool Offset
27. Maximum temperature offsets enabled by user during LOW TOU rates - Heat Offset
28. Maximum temperature offsets enabled by user during LOW TOU rates - Cool Offset
29. Maximum temperature offset enabled by user during DR Utility Events - Heat Offset
30. Maximum temperature offset enabled by user during DR Utility Events - Cool Offset
31. Settings for Savings (cPreferences.SelectedIndex)
32. OptIn to DR events (OptIn)
33. Demand Response Event (cDemandResponse);
34. Learn and Adapt (LearnAdapt)

APPENDIX D

Confidence Intervals

Different set points and/or pattern changes of the user, which represent the sample data for different occurrences observed during the simulation run. Let X_1, X_2,…, X_n represent the random sample occurrences of different patterns with normal distribution, and the mean value $\overline{X}$ is:

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \qquad \text{(D.1)}$$

The mean of each daily occurrences (pattern changes for a particular day of a week) during the simulation run, provides the single numerical value for the estimated expected value E|X| = μ. In order to verify how close the real values are to the estimate, we compute the variance $\sigma^{2:}$

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n} X_i - \overline{X} \qquad \text{(D.2)}$$

The small resulting values of σ^2 signify that the results are close to the mean value $\overline{X}$. Thus, we could be confident that $\overline{X}$ is close to the expected value E|X|. On the other hand, if the resulting value of σ^2 is large, it signifies that the results are widely dispersed around $\overline{X}$; hence, we cannot be confident that $\overline{X}$ is close to E|X|. Therefore, instead of seeking a single value to estimate the E|X|, we specify the confidence interval 100 (1 – α) %, which is highly likely to contain the true value of the parameter. Thus, we find the interval [L(X), U(X)] which define the limits of the expected value being within the chosen confidence interval. Thus, the probability P of μ being within the defined interval is defined as:

$$P = [L(X) \leq \mu \leq U(X)] = 1 - \alpha \qquad \text{(D.3)}$$

Where lower limit L(X) is defined as:

$$L(X) = \overline{X} - z_{\alpha/2}\frac{\sigma}{\sqrt{n}} \qquad \text{(D.4)}$$

Furthermore, upper limit U(X) is defined as:

$$U(X) = X + z_{\alpha/2} \frac{\sigma}{\sqrt{n}} \quad (D.5)$$

Thus, a 100(1 – α) % confidence interval for the mean μ of a normal population, when the value of standard deviation σ is known, is given by:

$$(\bar{X} - z_{\alpha/2} \frac{\sigma}{\sqrt{n}} \leq \mu \leq \bar{X} + z_{\alpha/2} \frac{\sigma}{\sqrt{n}}) \quad (D.6)$$

From the normal distribution table, choosing α = 0.05, we get the z value, $z_{0.05/2} = 1.96$. Therefore the 95 % confidence interval is:

$$(\bar{X} - 1.96 \frac{\sigma}{\sqrt{n}} \leq \mu \leq \bar{X} + 1.96 \frac{\sigma}{\sqrt{n}} = 0.95) \quad (D.7)$$

Where,

α = 0.05,

$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n} X_i - \overline{X}}$ is sample standard deviation,

σ^2 is variance,

$\bar{X}$ is sample mean, and

n is the sample size.

Thus, choosing α = 0.05, yields a 95% confidence interval. A confidence interval of 95% implies that 95 % of all the samples are within the interval that includes μ, and only 5 % of samples would yield erroneous interval. In this thesis, the confidence interval of 95% is used to validate the adapted values for changing patterns and/or user preferences, from the simulation runs.