Stony Brook University

NETWORKED ADAPTIVE CLASSIFICATION FOR CYBER-PHYSICAL SYSTEMS

A Dissertation Presented

by

Michael Gilberti

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

In

Electrical Engineering

Stony Brook University

December 2009

Stony Brook University

The Graduate School

Michael Gilberti

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation.

Dr. Alex Doboli, Advisor of Dissertation
Associate Professor, Department of Electrical and Computer Engineering

Dr. Milutin Stanacevic, Chairperson of Defense
Assistant Professor, Department of Electrical and Computer Engineering

Dr. Leon Shterengas, Assistant Professor
Department of Electrical and Computer Engineering

Dr. Edward H. Currie, Chief Information Officer
Tritium Technologies, Inc.

This dissertation is accepted by the Graduate School

Lawrence Martin
Dean of the Graduate School

ABSTRACT OF THE DISSERTATION

Networked Adaptive Classification for Cyber-Physical Systems

Michael Gilberti

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

2009

Cyber-physical Systems (CPS's) are networks of embedded systems which operate under tight power and timing constraints and with varying levels of precision. The primary goal is to meet the above requirements for an application that is distributed over a network.

Adaptive Classification is introduced in the context of a chemical gas cloud localization application running over the CPS. Standard machine learning binary classifiers are employed and optimized for implementation in reconfigurable hardware at varying precisions, power requirements and speeds.

An exemplar set of classifiers contains one chemical species classifier and four individual chemical classifiers such as, Alcohol versus the-rest (of the species) and methyl alcohol versus the-rest(of the individual alcohols). Other sets classify additional species and chemicals.

The network is populated with these classifiers by assigning one set of chemical classifiers per node and apply techniques from the field of evolutionary game theory to allow the nodes to self-configure over time.

Contents

List of Figures

List of Tables

ACKNOWLEDGMENTS

I am grateful to and thank my Ph.D. advisor Dr. Alex Doboli for his support, advice and interesting discussions on this topic. I want to thank my thesis committee, Dr. Alex Doboli, Dr. Milutin Stanacevic, Dr. Leon Shterengas and Dr. Edward Currie for their valuable feedback. Also, I would like to thank my wife, Geetika and my daughter Gianna for their patience, encouragement, and endless support during this journey. I want to also thank the Gilberti family and the Puri family for their support.

Chapter 1

Introduction

Concentrations of volatile organic compounds (VOC's) are dangerous to people and a hazard to the environment. High concentrations of VOC's can ignite and explode, while smaller concentrations can cause respiratory and/or kidney failure. One popular method to detect VOC's is known as Ion Mobility Spectrometry (IMS). IMS sensors allow real-time monitoring of VOC's chemical warfare agents, explosives and narcotics [1]. IMS sensors are currently used to monitor VOC's on-board the international space station, as well as, to detect subsurface VOC's like MTBE, a gasoline additive, in the soil between the surface and ground water. A network of adaptive classifiers capable of detecting and localizing airborne VOC clouds within an area covered by the network is described. Fast detection and localization of VOC clouds can lead to quick evacuation of people, immediate shut-down of potential sources of ignition and fast/accurate deployment of first responders (e.g. Fire and Rescue units). Thereby making it possible to save lives and preserve infrastructure.

Five classes of VOC's for detection, each containing four chemicals are identified.

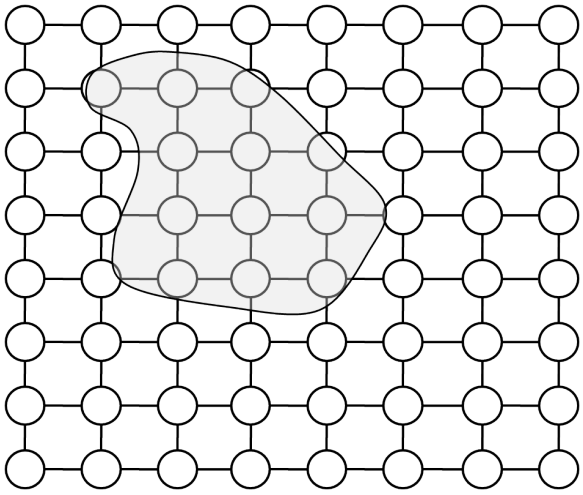


Figure 1.1: A Network of Adaptive Classifiers with Chemical Cloud

The network classifies and localizes VOC clouds by reconfiguring each of its nodes. Each node consists of a miniature ion mobility spectrometer (IMS) for acquiring chemical data from the environment, a microcontroller, a field-programmable gate array device (FPGA), memory and a power source (battery). The microcontroller communicates with other nodes; controls the FPGA and IMS; and monitors the available power. Adaptive classifier(s) are contained within the reconfigurable logic of the FPGA and used for classifying the data sensed by the IMS. Memory is used to hold multiple adaptive classifier configurations. The IMS is a system, not just a sensor. It has its' own control system independent of the microcontroller, FPGA and memory of the node architecture.

An Adaptive classifier is a machine learning algorithm implemented in hardware optimized using a variable bit-width design flow. Each classifier is given a set of valid bit lengths (widths) and a cost function. Simulated Annealing is used to reduce the overall required hardware/power while maintaining a certain level of classification

accuracy. This is done by exchanging a bit width at random, running the classifier, computing the result and accepting, or rejecting, the change based on a function of how many iterations of the algorithm are required for completion. Using this technique it has been demonstrated that a 25 percent reduction in area/power can be achieved.

Each node has a set of binary classifiers. One classifier that allows it to discriminate between one chemical species (class) versus the-rest of the possible species (classes) and a group of One-versus the-Rest classifiers that allow a node to identify individual chemicals (analytes), with the condition that they belong to the same class as the chemical species classifier . For example, if the node has a set of classifiers that contain an Alcohol-versus-the-Rest classifier then it also contains the following alcohol classifiers; methyl versus the-rest, ethyl versus the rest, propyl versus the-rest and isopropyl versus the-rest. The number of analyte classifiers was restricted to four, however, the number could be higher as there are many more alcohols within the species.

Each classifier comes in three bit width configurations 12-bit max, 10-bit max and 8-bit max. Each configuration corresponds to a mean-squared-error (MSE) that defines the accuracy of the classifier. In addition, the number of multipliers within these configurations may be varied to allow between 2^0 and 2^8. Varying the number of multipliers in use at a given time has the effect of reducing the area used and the power consumption over time, as well as, the speed of computation. The set of possible configurations comprise a set of three Pareto-optimal design-point curves. These Pareto curves define the customization options for the classifier. This works in

combination with a set of reconfiguration rules designed to match the configuration to the available computational resources at both the node and neighborhood levels. Chapter 3 describes the adaptive classifier approach.

The principals of evolutionary game theory are applied with the goal of supplying the correct classifiers to each local set of nodes, over the event time. It is shown that by doing so enables the network to localize clouds of chemicals, if they are present, in any of the local neighborhoods.

Since the network of nodes has a rigid grid structure a spatial game is employed. A spatial game is one where nodes cannot meet randomly but are confined to play its games with surrounding nodes. The surrounding nodes form a nodes neighborhood. The game used for all of the experiments is the Prisoners' Dilemma. It is widely used by game theorists and evolutionary biologists, economists and political/social scientists. In the context of a network of adaptive classifiers, it is used to propagate the best classifiers to nodes that may need them, in order to track and localize the chemical cloud(s). Chapter 4 is dedicated to the network of adaptive classifiers and describes its operation in detail. Chapter 5 describes the experiments and provides a discussion of the results.

Cyber-physical system (CPS) refers to the tight conjoining of and coordination between computational and physical resources [2]. The National Science Foundation believes it to be a key area of research and foresees that future cyber-physical systems will be more adaptable, autonomous, efficient and functional than present day embedded systems. The Networked Adaptive Classifier approach to cyber-physical systems presented herein aims to meet NSF's vision for such systems.

Chapter 2

Related Work

2.1 Evolutionary Game Theory

Game Theory was invented by John von Neumann and Oskar Morgenstern in 1944[3]. Evolutionary Game Theory first appeared in the early 1970's [4][5]. The prisoner's dilemma, to be described later in this text was invented around 1950 by Merrill Flood and Melvin Dresher [6]. Robert Axelrod [7], in his best selling The Evolution of Co-operation describes the mechanisms behind how cooperation between two parties can emerge in different situations and lead to a more beneficial outcome than competition alone. He conducted two computer prisoner's dilemma tournaments in the early 1980's to try and discover the best strategy to use in an iterative prisoner's dilemma game, if indeed one existed. His conclusion was that the strategy of Tit-for-Tat was the best. The premise behind the strategy is to begin with cooperation but then do what the opponent did on the last round of game play. This simple strategy promotes cooperation while punishing non-cooperators, otherwise known as defectors.

Nowak [8] uses evolutionary game theory to gain new insight into cancer and the HIV virus. He describes a strategy that beats Tit-For-Tat, called Win-Stay-Lose-Shift.

Nowak, Hauert and Langer [9] explore the prisoner's dilemma when it is played as a spatial game where players no longer meet randomly within a population, but are constrained to play the game repeatedly with their nearest-neighbors. Through experimentation they found a set of conditions in which cooperation can invade, survive and thrive in a population of all defectors. They found that cooperator must form compact clusters in order to survive, the minimum size of which is size a 3 x 3 cluster of cooperators. Moreover, they found when a small cost to benefit ratio (c/b = 0.02) is used in an iterative prisoner's dilemma, cooperators form a large densely packed cluster, with few defectors remaining within. In addition, when c/b is closer to the extinction threshold (c/b = 0.12) a much less dense cluster forms with many large pockets of defectors.

Axelrod [10] and Mittal and Deb [11] use genetic algorithms to find optimal strategies of the iterated prisoner's dilemma. Axelrod searched for a strategy that maximizes its' own score. Mittal and Deb take a multi-objective approach and try to find a strategy that will maximize a players' score while minimizing the opponents score.

Evolutionary game theory is also being applied to sensor networks and to smart energy grids. In [12] cooperation is explored for routing communications in underwater sensor networks governed by two different central authorities. The sensor nodes play the prisoner's dilemma to decide whether or not to forward information packets of the other networks' nodes to communication buoys. Hines and Talukdar [13] de-

scribe reciprocally altruistic agents for the mitigation of cascading failures in power grids, an approach to avoiding massive blackouts of service. They sense the capacitive load on power station to determine whether neighboring power stations will cooperate to receive the excess load.

2.2 Binary Classification

Brooks and Iagnemma [14] describe a novelty detector that visually detects new terrain from terrain for which a mobile robot has already been trained. It uses the two-class, classification methodology and support vector machines to improve navigation in a simulated Mars environment. Hill and Doucet [15], devise a geometric construct that aides in applying the two-class support vector machine to multi-class classification. Ayech and Trabelsi [16] show how to decompose a multi-class problem into a series of two-class classification problems using neural networks. Evan-Zohar and Roth [17] use their Sequential Model similar to a decision tree, to formulate the multi-class classification problem for natural language processing in terms of two-class classification. Their model sequentially restricts the set of possible classes to a small set, in a data dependent way.

2.3 Gas Classification

Bermak and Belhouari [18] apply Bayesian Learning to the gas identification problem and shows how it can outperform K-Nearest Neighbor and multilayer perceptron in

this task. They combine the Gaussian process with principal component analysis and use tin-oxide gas sensors in their work.

Shi and Bermak et al. [19] developed an electronic nose consisting of an array tin-oxide micro-hotplates and a dynamically reconfigurable FPGA capable of identifying five different combustible gases with two in various concentrations. They use a committee machine (CM) classifier, which combines the outputs of five other classifiers to obtain a decision with improved performance over any single method.

Before input into the CM, confidence transform (CT) functions are first applied to the results of both an MLP and Radial Basis Function neural network and likewise to the Gaussian Mixture Model, k-nearest neighbor and Probabilistic Principle Component Analysis classifiers. In addition, they propose using a weighted combination rule to prevent any single poorly performing classifier from affecting the CM decision.

Principle Component Analysis (PCA) is used to preprocess the gas datasets in order to project them onto a lower dimension. PCA is implemented in the first of three hardware configuration stages after data acquisition. It is implemented as the vector-matrix multiplication $z = xT$, where z, x and T are the transformed pattern, the original pattern and the transform matrix, respectively. PCA is implemented in a hardware circuit that is based on Distributed Arithmetic (DA) which is an algorithm that performs this computation using pre-computed lookup (LUT's) tables instead of logic.

Average classification results were provided for principle components ranging from 2 through 8. The CM had the best classification rate (95.9%) with 8 principle components and performed worst (82.7%) when the data was reduced to only 2. This is not

surprising as one would expect performance to decline linearly with fewer principle components. In all cases, the CM had a higher classification rate than any single classifier.

The slow response of the gas sensors was exploited by allowing the system to be partitioned into several stages of computation, i) sampling and preprocessing, ii) CM, and iii) CT and decision, which were implemented sequentially using the multiplexing capability of the Xilinx Virtex II FPGA. This device, along with the present generation, can dynamically switch between a maximum of five configurations all of which are stored in SRAM. This is useful if the hardware resources required exceed those available on the chip.

In the first stage the sensor output is digitized using an analog MUX and 12-bit serial ADC. At this point the FPGA is configured to implement four circuits for the detection of a steady-state, normalization, PCA and control. The control circuit generates signals to control the analog MUX and the other three circuits of the FPGA along with generating clock signals for the ADC.

The second stage implements the five classifiers which execute in parallel. The KNN consists of five each of ROM's, subtraction and square units, and four adders. Three comparators and three registers (for K=3) are used to find the nearest neighbor in winner take all fashion. In the MLP, the dot product of weights and inputs is implemented by means of DA and since there are six hidden nodes and 5 output nodes the implementation uses eleven DA units and eleven adders, one for each node of the network. The neuron's activation function is implemented using a piecewise linear function (LPF) approximation. The RBF neural network could not be implemented

in DA as there are too many hidden nodes (13 nodes) for it to run in parallel therefore a more conventional computational structure was chosen that uses a multiplier, six accumulators, a subtraction unit, LPF and square units. Both GMM and PPCA are implemented using a serial parallel vector matrix multiplier, a square and multiplier units and LPF to approximate the exponential function. The differences between the two are their parameters are stored in ROM.

The third stage applies the five CT's and final decision function in a winner take all circuit based on a weighted majority vote function. The CT for KNN provides 5 bit labels of the 3 nearest neighbors that control multiplexers to decide whether or not to increase the confidence of each class. It consists of 5 accumulators and 5 multiplexers. The CT's for the other four classifiers just normalize the output to a value between 0 and 1 in order to approximate the probability of each class. The final decision circuit is comprised of five multipliers an adder and the winner-take-all comparison unit.

FPGA implementation results show that system ran at a clock speed of 50 MHz and the circuits executed in 26 ms. The Data acquisition and signal processing stage used the least amount of chip resources requiring only 3,812 4-input LUT's and 2,723 slices, 13% and 19% of those respective resources. On the other-hand, the classifiers consumed the most resources, 20,115 4-input LUT's and 12,145 slices, accounting for 70% and 84% of those respective resources. CT and Decision circuits were implemented in 4,236 (15%) LUT's and 3,075 (21%) slices. These results show that these circuits cannot all be mapped to a single FPGA at the same time and therefore could not be implemented without using dynamic reconfiguration or moving

the design to a larger device.

2.4 Classification of Mobility Spectra

Ion mobility spectrometry (IMS) and differential mobility spectrometry (DMS) are two technologies in use today for the detection and classification of chemicals. IMS measures the time it takes a chemical ion to move through a uniform, low DC electric field, whereas DMS measures the velocity of an ion type in both low and high electric fields [20][21]. Hartman, et.al [22] presents an embedded microcontroller for control and signal processing for an IMS sensor.

Bell, et al. [23] developed a database of 1293 ion mobility spectra based on mV intensity and applied a two-tiered (hierarchical) MLP approach to classify 195 chemicals of various concentrations, first into one of 10 classes, and then to identify the specific chemical. After careful optimization of the database the system was able to successfully identify 91% of the spectra of an independent test set consisting of 15% of the spectra randomly selected from the database, but not used for training. Optimization of the database was aided by the neural network response which identified possible deficiencies with a spectrum such as low concentration or spectral distortion due to the presence of impurities. The suspect spectra were pruned from the database helping to improve the classification rate.

The first MLP used the back-propagation of error training algorithm and consisted of 100 hidden nodes and 10 output nodes, one output node per chemical class.

The second MLP also used back-propagation and had 100 hidden nodes but used

the number of output nodes corresponding to the number of chemicals assigned to a particular class. For instance the alkanes class has 14 members so the MLP for identifying alkanes has 14 output nodes. After first attempts using normalized input data had failed to produce satisfactory results, the data for both neural networks was pre-processed using a log transformation in order to conserve spectral information (i.e. ion drift time and peak shape) while attenuating peak height. The authors made no mention of which activation function was utilized to squash" the outputs of the network nodes. The MLP was found to be able to extract several spectral features and use them for classification. These included drift times, number and intensity of peaks and peak shape. Not all networks used the same features but rather used those that could be isolated and distinguished from other spectra in an opportunistic manner.

Eiceman et al. [24] used a similar MLP approach to classify differential ion mobility spectra of a number of chemicals classes such as alcohols, ketones, substituted ketones and aromatics. Between 20 and 41 spectra from each class was used for training the MLP. The performance of the neural network on trained (familiar) data ranges between an average classification rate of 76% for ketones to 98.4% for substituted ketones.

When presented with unfamiliar data, which had not been used for training, the network was able to successfully classify between 81% and 99.1% of the spectra, with the overall average rate being 91.5%. In general, classification of DMS data is better on most spectra than data acquired by IMS techniques, the reason for which is not well conveyed.

2.5 Bit-width Minimization

This section explores the various techniques that enable reconfigurable computing devices to compute using fixed-point operation of variable precision.

2.5.1 BitSize

Gaffar, Mencer, Luk and Cheung [25] developed the BitSize tool for optimization of both floating point and fixed point designs that explores a design's accuracy, dynamic range, area and speed. The Bitsize tool accepts a design description in either C/C++ or Xilinx System Generator code and outputs an annotated data flow graph. Fixed-point designs are then verified and implemented using Xilinx System Generator while floating-point designs are simulated and implemented using parameterized floating point hardware library.

The problem of minimizing the bit-widths is divided into range and precision analysis where range deals with the integer (exponent) bit-widths while precision part minimizes the fraction (mantissa) of the fixed-point (floating-point) number.

A technique developed by the applied mathematics community known as automatic differentiation [26] is employed to calculate the bit-widths. It is a way of differentiating an algorithm and doing so during the algorithms execution with little change to the original code. It is fast, requiring only one iteration and the method allows for multiple bit-width formats as opposed to only uniform bit-widths. At each operator node of a data flow graph automatic differentiation will calculate the gradients of the outputs with respect to the inputs then annotates the graph edges with the

associated gradient. This describes a sensitivity relationship between the outputs and inputs such that an output y is a function of an input $x1$. A change $\Delta x1$ occurring in the input $x1$ causes a change Δy in the output y.

The BitSize algorithm begins with the user applying constraints based on error, dynamic range, area and speed. Next, range analysis is accomplished by observing the values passing though the nodes of the data flow graph of the un-optimized design and determining the dynamic range at each node. Precision analysis uses automatic differentiation to determine the maximum error tolerance at each node. Next, an area estimator calculates the area usage of the bit-width optimized design based on an area model. In addition there is a performance data model that contains area and speed models for the operator blocks. Both the area estimator and performance model are used to decide the most suitable data type for implementation, fixed or floating-point. The tool provides several back-ends for targeting different implementations.

BitSize was used in four case studies including ray tracing, function approximation, finite impulse response (FIR) filtering and discrete cosine transform (DCT), all of which were implemented using Xilinx Virtex 2 FPGA's. In the case of ray tracing the fixed-point implementation uses 40% fewer resources than a floating-point implementation when considering similar output error. The fixed-point design operates at 100MHz while the floating-point is slower at 80MHz. The method is used to indicate the point at which to switch over to a floating-point design which in this case is when a dynamic range of 107 is required. When considering function approximation the resource reduction was between 15% and 20% sometime favoring floating-point over fixed-point. Since FIR filtering requires a large dynamic range, up to 1016 it favors a

floating-point implementation that uses 30% less area as compared to the fixed-point design. Lastly, for the DCT case study fixed-point used only 10% of the LUT of the FPGA than did the floating-point design, however the latter used far fewer embedded multipliers (25%) so the user has to decide whether to optimize around LUT's or embedded multipliers.

2.5.2 MiniBit

MiniBit is an automated approach to fixed-point bit-width optimization proposed by Lee, Gaffar, Mencer and Luk [27] that uses static analysis via Affine Arithmetic. According to the authors they chose static analysis over dynamic analysis (where stimuli input signals are applied as in Bitsize Tool) to avoid long simulation times even though dynamic analysis may provide results that are closer to optimal. Static analysis applies the characteristics of the input signals rather than exemplar signals. Affine Arithmetic is used to optimize both the range and precision of the fixed-point numbers.

Affine Arithmetic is considered to be an improvement over previous efforts that made use of Interval Arithmetic which is a mathematical technique that seeks to bound rounding errors. The difference between the two is that Affine Arithmetic uses a separate uncertainty parameter for each signal. Whereas many bit-width optimization techniques use signal-to-noise ratio (SNR) as a quantization error metric, MiniBit uses an error metric called unit-in-the-last place (ulp) to analyze precision and results are accurate to 1 ulp. While SNR is popular with DSP applications mainly because it

is portable between modules of large designs it does not provide a maximum absolute error bound as is the case with ulp. It is described as follows: if the fractional part of the fixed-point number has 8 bits then the ulp would be 28 and if it has 10 bits then the ulp would be 210, etc. This means that the maximum absolute error is less than or equal to 28 and 210 respectively. In addition, MiniBit determines both an analytically derived uniform fraction bit-width as well as a multiple fraction bit-width found through adaptive simulated annealing, enabling the design of multiple wordlength systems.

Range and precision optimization are performed separately again with the aim of minimizing the number of bits for the integer part (range) and fractional part (precision) of the fixed point numbers. MiniBit is built on the authors' previous work with the BitSize bit-width analysis system [28]. It takes as its' input a high-level description such as C/C++ or Xilinx System Generator code along with an error requirement and outputs a cost function and an error function.

Range analysis is performed (via Affine Arithmetic) first and the results are passed along for precision analysis which operates in two stages. The first uses the error function generated by MiniBit to find the optimum uniform fraction bit-width which serves as the initial starting point for adaptive simulated annealing which finds the optimal multiple fraction bit-widths using the cost function and error constraints.

MiniBit was implemented on five case studies and multiple bit-width implementations were compared with their uniform bit-width counterparts. For degree four polynomial approximation, RBG to YCbCr conversion, 2 x 2 Matrix Multiplier and 8 x 8 discrete cosine transform implementations, area and latency improved between

2% and 10%. The B-spline implementation saw the greatest benefit with a nearly 20% improvement in area and 12% improvement in latency. In addition, both maximum and average ulp errors were below the target of one ulp and the SNR ranged between 50 dB and 150 dB.

2.5.3 PowerBit

PowerBit, a power aware bit-width minimization technique by Gaffar, Clarke and Constantinides [29] leverages off of both the BitSize tool and MiniBit technique but seeks to minimize the bit-width through evaluation on a power cost function rather than only relying on reducing power as a side effect of area reduction.

PowerBit uses the BitSize tool to generate the power cost function and error function both of which are then used to find the optimal range of the fixed point number by way of Affine Arithmetic. The results of the range optimization are then used to analyze the precision (fractional bit-width) and to determine the uniform fractional width (i.e. the single fraction-width that is best for the system). As in MiniBit adaptive simulated annealing and the error function are used to find the multiple fraction width, but this time the power cost function is employed as opposed to one that includes only area and latency. The BitSize back-end stage converts the optimized design description into a hardware implementation via VHDL.

Power consumption is calculated at the block level only after placement, routing and simulation. Switching activity level obtained from simulation are then fed into the Xilinx XPower tool to estimate the total power consumed.

The authors present four example implementations. The first three; cascaded addition, 2 x 2 matrix multiplication and uniform cubic B-spline, demonstrate over a 10% reduction in logic power for multiple fraction bit-widths, while the fourth, an 8 x 8 discrete cosine transform exhibited a 20% percent reduction. They show that area optimization alone can sometimes lead to an increase in power consumption as was the case with the cascaded addition and 2 x 2 matrix multiplication examples, where they exhibited a 2% and 8% increase in power consumption respectively.

Interestingly, PowerBit was able to optimize components of the same type individually, based on location within the data flow graph, while area based optimization could not, the reason for which is not entirely clear and one could speculate that it a side effect of the modeling technique.

2.5.4 Word Length Conscious High Level Synthesis

Kum and Sung [30] introduced a combined fixed-point word-length optimization and high-level synthesis design flow for designing digital signal processing systems. They present a more accurate cost function by tracking the amount of hardware sharing of the functional units (i.e. adders and multipliers) and through signal grouping. They also reduce optimization time significantly.

First, minimum word-lengths are determined through simulations of a signal flow graph and then grouped and assigned to hardware. A cost function is formulated and minimized using both list scheduling and integer linear programming algorithms for scheduling operations and binding to the hardware.

In their list scheduler, ready lists are generated for hardware units having different word-lengths. The one with the longest word-length gets the highest priority. Since the list scheduler is close to, but not optimum, they use ILP as a secondary step. ILP is implemented using a data flow graph $G(V, E$ having $n(|V|)$ operations and $n(|E|)$ data dependencies and minimizes a cost function that is the sum of the number of functional unit types multiplied by their cost under three constraints. These constraints are related to the number of functional units of a given type and number of operations allowed per control step as well as the order of operations. A modified clique partitioning algorithm is used as the basis for register binding by iteratively searching for and deleting the node with the maximum word-length of the graph until the number of vertices is zero.

The design flow was used to generate several designs including a fourth order infinite impulse response (IIR) filter with 15% hardware reduction, a fifth-order elliptic filter with 7% hardware reduction and a 12th order adaptive least mean square (LMS) filter which benefited from hardware sharing information that reduced from 15 to 6 the number of signal groups.

Chapter 3

Adaptive Classifiers

3.1 Introduction

This chapter presents parallel implementations of the multilayer perceptron (MLP) neural network classifier in fixed-point reconfigurable hardware with adaptive precision. The MLP was trained off-line to classify chemical species using the standard back-propagation algorithm. The chemical data used for training was obtained from a pre-existing database that relates chemical species to data collected from an ion mobility spectrometry (IMS) sensing device.

IMS sensors allow real-time monitoring of volatile organic compounds (VOC's), chemical warfare agents, explosives and narcotics [1]. IMS systems are currently used to monitor airborne VOC's on board the International space station and to detect subsurface VOC's, e.g. MTBE in the soil between the surface and groundwater [31].

The final parallel implementations of the MLP employs computing blocks with variable bit widths. They include 260 multipliers with bit widths ranging from 10-

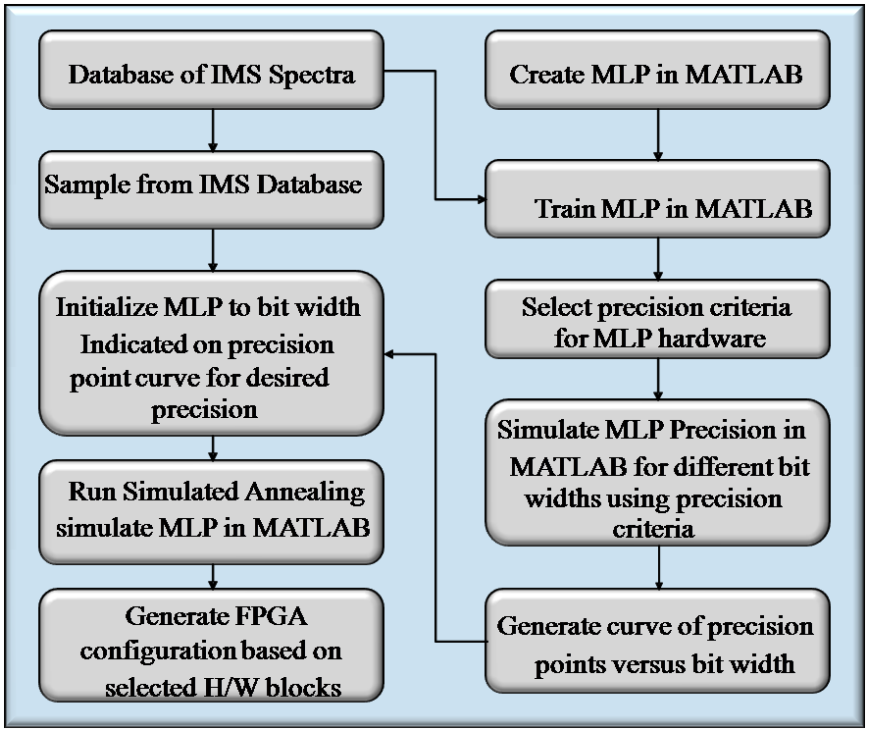


Figure 3.1: Design Flow

bits down to 6-bits and 260 adders with bit widths that are two times as long as each multiplier bit width. The designs were generated as a result of optimizing two initial configurations comprised of all 10-bit (8-bit) multipliers and 20-bit (16-bit) adders. This is performed by means of simulated annealing (SA) which optimizes a cost function that includes the number of hardware units required for the circuit and the amount of error produced as a result of the MLP computations. Design flow as described previously in [32] and shown in figure 3.1 is applied but changed from floating-point to fixed-point hardware.

The hardware unit selected was the Xilinx *slice* which has a ratio of approximately 1:2 with a Xilinx Look-up Table (LUT). The error term used was the sum squared of the error between the 64-bit MLP results and the reduced bit width results obtained

after computation with the lower bit width hardware blocks.

Hardware Synthesis and Matlab simulation results show that the final implementation resulting from SA optimization uses considerably less hardware and provides the same level of error as the initial configuration. In addition, the final implementation also dissipates less power than the un-optimized initial design. It is also important to note that no change was made in the number of computing blocks between the final implementation and initial configuration. Only the bit width sizes were altered.

The reason for a parallel implementation, as opposed to one that is serial, is to allow construction of a multi-species classifier from a two-species classifier by multiplexing each MLP's weight set and comparing, and then accumulating the outputs.

For example, suppose that the task is to classify three species of chemicals species A, B and C. In a parallel design all the weights of an MLP trained to classify species A and B are multiplied during the same clock cycle. During the next clock cycle the weights are multiplexed to form a species A and C classifier and multiplexed again during the third clock cycle to form a species B and C classifier. The species whose accumulated outputs total 2 is the class that best matches the input to the MLP. Whereas previous research on classification focused on using one or two large MLP's having many nodes, this work focused on developing MLP's with small numbers of nodes with each MLP responsible for a small comparison (e.g., in this case only two classes of VOC chemicals, alcohols and aldehydes are compared). The goal is to have many small MLP's, either pipelined or running in parallel, and then aggregate the results for the classification task. This approach is expected to yield more robust classifiers that are suitable for hardware implementation.

3.2 MLP Classification of Chemical Data

Due to the similar appearance between the signals of each class, the MLP were trained to discriminate first between alcohols and aldehydes The similarity in appearance (max. peak at 170 ms) suggested that the two classes would be more difficult to classify over two sets of more dissimilar signals (ref. Figure 1). 4 of the 27 Alcohols and 4 of the 18 Aldehydes contained within the database were randomly chosen for inclusion in the training data. Eight signals from each individual chemical were used in training for a total of 64 training vectors

The 64 vectors were broken-down into three sets. One set was used for training the MLP, the second for testing the MLP and the third for validating the results.

The structure of the MLP consists of 128 inputs, 2 hidden nodes and two output nodes. Each hidden node multiplies every input by a weight, sums the results and adds a bias factor. The tanh sigmoid function is applied to the resulting summation. These results, scaled between -1 and +1 by the tanh function, are multiplied by the weights of each output node, summed, and a bias factor added. The tanh sigmoid function is again applied resulting in the output of the MLP. A value of approximately +1 for node one and approximately -1 for node two signifies that the input signal was classified as an alcohol while a value of approximately -1 for node one and approximately +1 for node 2 signifies that the input signal was classified as an Aldehyde.

3.3 Parallel Hardware Implementation

Parallel hardware implementation implies that there exist two multiply-add trees, one for each hidden node, running in parallel. This is followed by two smaller multiply-add trees, one for each output node, that also run in parallel. The multiply-add trees are comprised of 128 multipliers and eight levels of adders totaling, 128 adders for the hidden nodes. The output multiply-add trees consist of 2 multipliers and 2 adders with two levels of adders.

3.4 Single Bit Width Design

The hardware slice usage for the MLP described in section 3 was estimated using only one bit width type of the N-bit multiplier and 2N-bit adder.

Table 3.1 shows the total number of hardware slices for each design. It also shows the mean of the sum squared of the errors taken for all 64 training, test and validation data vectors at a particular precision. The individual errors are the deviations from the 64-bit double precision floating point outputs computed in Matlab. These were the original results of training the MLP in Matlab. The last column lists the number of misclassifications produced after running the complete set of 64 training, test and validation vectors through the particular N-bit design. It shows that the configurations using only 7 or 6-bit multipliers with 14 or 12-bit adders, misclassify some of the data whereas higher bit width configurations do not. This is an indication of the classification accuracy of a particular MLP.

Table 3.2 shows the post-synthesis results for the design using all 10-bit multipliers

Table 3.1: size, error and accuracy by bit-width

Multipliers / Adders	Total Slices	Mean SSE	Misclassifications
10/20-bits	17,420	0.016199	0
9/18-bits	16,900	0.0272	0
8/16-bits	11,700	0.094394	0
7/14-bits	10,920	0.226207	2
6/12-bits	7,020	0.742147	15

and 20-bit adders. Results include the added slices for the tanh activation functions, weight and input registers.

Table 3.2: all 10-bit post-synthesis results

Slices	LUT's	FF's	Power
19,470	34,635	6,564	1361 mw

3.5 Variable Bit Width Design

Simulated annealing was used to generate a variable bit width design that is optimized around hardware slice usage and output error. Beginning with an initial configuration comprised of all 10-bit multipliers and 20- bit adders, the algorithm used only two of the 64 data vectors, one each for the chemical species alcohols and aldehydes.

Results are provided for three simulations in Table 3.3. The first uses 4,454 fewer hardware slices than the all 10/20-bit design of Table 3.2 and yet has a slightly better mean SSE with no misclassifications A plot of the error and hardware usage for the

simulation run is shown in Figure 3.2. The error shown in the Figure is the mean value of sum squared of the errors of the output nodes for data vectors D1 and D33. In this case the best error occurred at iteration 878 when the error was 0.0014105 and the hardware usage was 12,966 slices.

Table 3.3: size, error and accuracy post-optimization

Multipliers	Total Slices	Mean SSE	Misclassifications
10,9,8,7 & 6-bit (variable)	12,996	0.016116	0
8, 7 & 6 bit (variable)	11,312	0.080156	0
8, 7 & 6 bit (variable)	9,322	0.171723	0

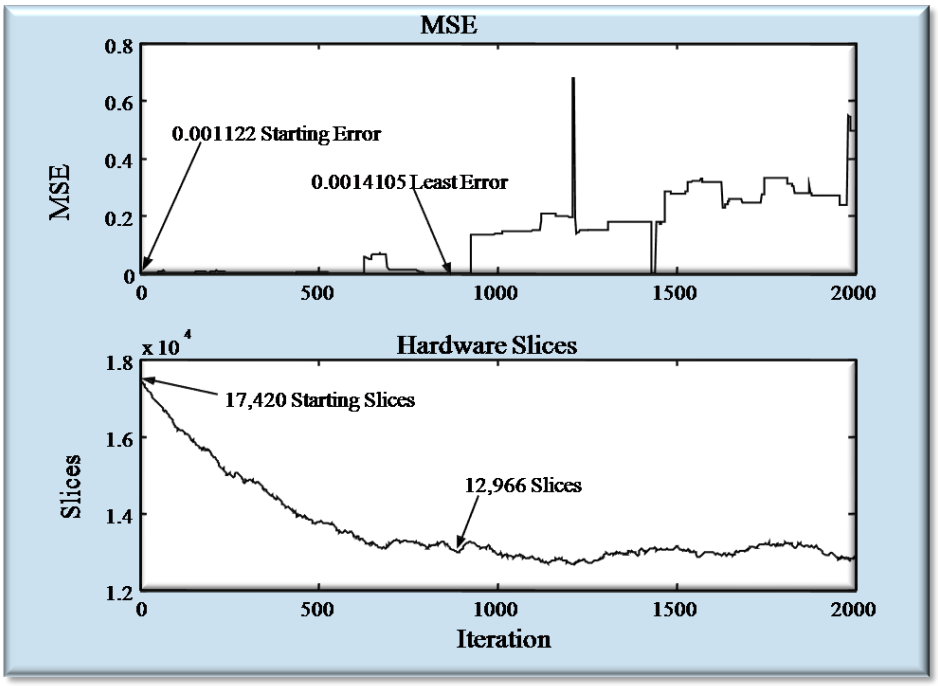


Figure 3.2: simulation run for 10, 9, 8, 7 6-bit variable bit width design

Table 3.4 shows the responses to alcohol and aldehyde input spectra of one training, validation and test (TVT) vector selected out of the set of 64. The first and

second group are the results with all 10 bit multipliers and 20 bit adders while the third and fourth groups are post optimization results with 10, 9, 8, 7 and 6-bit multipliers and 20, 18, 16,14 and 12-bit adders.

Tables 3.5 and 3.6 show the number of multipliers and adders for given bit widths before and after optimization. If a device is chosen that has embedded multipliers, as most now do, this table can be used to choose which multipliers to move from a logic implementation to embedded multiplier. This would further reduce the number of slices required of all designs.

3.6 Power Dissipation Estimates

It should be noted that any reduction in the number of hardware slices reduces the power consumed by the device. Table 3.7 provides estimated power dissipation based on the slice usage estimates. Power dissipation was calculated using the Xilinx web-based power calculator for Virtex II Pro devices.

3.7 Adaptive Reconfiguration

Device reconfiguration is performed through the selection of Pareto-optimal design points from the design point curve of Figure 3.3. The curve contains design points from both the single and variable bit width designs presented in section 4. Note that the design point containing 11,700 slices is not a Pareto point and can be eliminated. Three scenarios are presented for which the ability to adapt the hardware is

Table 3.4: 10-bit TVT data vector responses

Alcohol Node		Aldehyde Node		SSE
64-bit	10bit	64-bit	10-bit	
Pre-Optimization Alcohol Responses				
0.96391	0.96332	-0.98192	-0.999	0.001122
0.96198	0.93329	-0.98093	-0.999	0.002376
0.93258	0.78448	-0.96077	-0.82959	0.020561
Pre-Optimization Aldehyde Responses				
-0.96101	-0.999	0.97645	0.999	0.001122
-0.94798	-0.999	0.96738	0.999	0.002376
-0.96047	-0.999	0.97671	0.999	0.020561
Post Optimization Alcohol Responses				
0.96391	0.93988	-0.98192	-0.999	0.001411
0.96198	0.72968	-0.98093	-0.95032	0.029253
0.93258	0.99628	-0.96077	-0.999	0.00375
Post Optimization Aldehyde Responses				
-0.96101	-0.999	0.97645	0.999	0.001411
-0.94798	-0.999	0.96738	0.999	0.029253
-0.96047	-0.999	0.97671	0.999	0.00375

Table 3.5: number and bitwidth of multipliers after optimization

Mults	10-bit Only	10-bit & less	8-bit Only	8-bit & less #1	8-bit & less #2
10-bit	260	57	0	0	0
9-bit	0	55	0	0	0
8-bit	0	52	260	47	232
7-bit	0	41	0	97	17
6-bit	0	55	0	116	11
Total	260	260	260	260	260

Table 3.6: number and bitwidth of adders after optimization

Adders	20-bit Only	20-bit & less	16-bit Only	16-bit & less #1	16-bit & less #2
20-bit	260	69	0	0	0
18-bit	0	43	0	0	0
16-bit	0	53	260	36	138
14-bit	0	52	0	120	66
12-bit	0	43	0	104	56
Total	260	260	260	260	260

Table 3.7: area and power dissipation of multipliers and adders

Multipliers / Adders	Total Slices	Estimated Power Dissipation
10/20-bits	17,420	1296 mw
9/18-bits	16,900	1272 mw
10/20, 9/18, 8/16, 7/14 & 6/12-bit (variable)	12,966	1091 mw
8/16-bits	11,700	1032 mw
8/16, 7/14 & 6/12 bit (variable)	11,312	1021 mw
7/14-bits	10,920	996 mw
8/16, 7/14 & 6/12 bit (variable)	9,322	922 mw
6/12-bits	7,020	816 mw

advantageous.

The device has the ability to dynamically switch between the loaded configuration and any other that may reside in its, memory, otherwise known as its configuration space. For this reason the three optimized classifiers are stored at unique addresses in the configuration space and the device can multiplex between them. It takes 79 milliseconds for the device to switch between configurations. The old configuration is erased from the device and the new configuration is loaded.

3.8 Synergistic Reconfiguration

Classifying IMS spectra into one of two chemical species is a small part of the overall scheme. Periodically, it is desirable to identify the individual chemical and this requires that an additional classifier be employed. One way to do this is to reconfigure the device to add this functionality. In doing so, it would be advantageous to keep

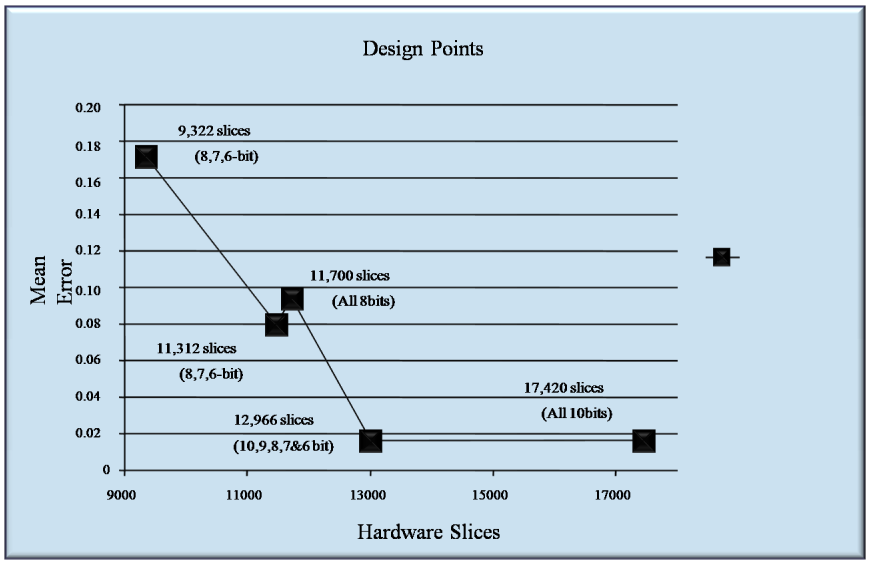


Figure 3.3: Design Points

the chemical species classifier in memory for the purpose of change detection, but switch to a smaller design by selecting an appropriate design point. In this way, two (or more) classifiers could operate simultaneously.

With this in mind one can imagine a scenario where the highest precision chemical species classifier is loaded in configuration 1. This corresponds to design point with 17,420 slices of Figure 3.3, or perhaps to an optimized 12-bit, or higher precision classifier Upon positive classification of an alcohol compound the device is reconfigured to load a high precision individual alcohol classifier selected from a comparable design point curve (not shown). Next, if the alcohol classifier identifies the alcohol as being isopropyl alcohol, this positive classification and identification allows the device to be reconfigured to a third configuration were the chemical species classifier having 9,322 slices is selected from the set of Pareto points on the design curve of Figure 3.3. Similarly, an individual alcohol classifier is selected from a comparable design

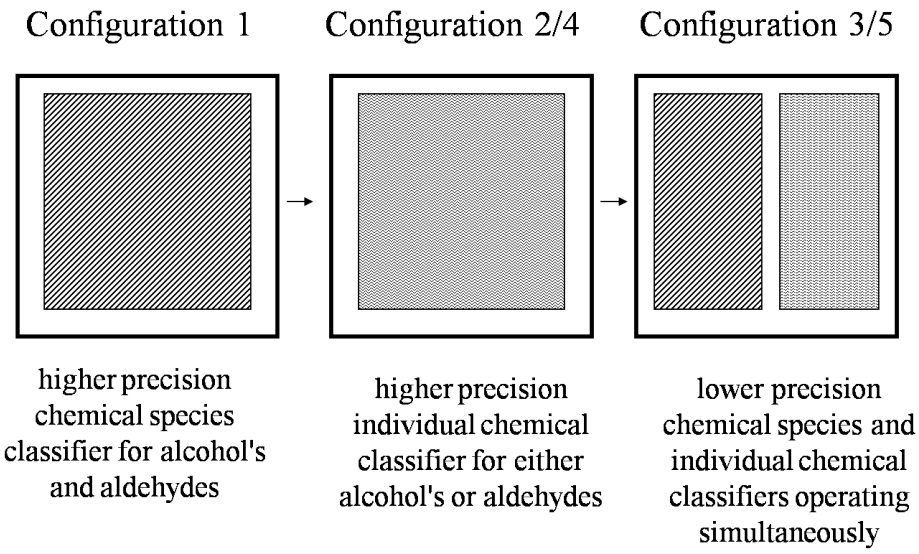


Figure 3.4: Synergistic Reconfiguration

point curve (not shown) and both are operated simultaneously. Figure 3.4 depicts this sequence.

This is advantageous because the individual chemical classifier has no facility for detecting if the IMS sensor begins sensing an aldehyde compound. If this were to happen the output of the individual alcohol would become meaningless and the device would switch back to configuration 1, then to a high precision aldehyde classifier, and finally to a fifth configuration comprised of the reduced precision species classifier and a reduced precision aldehyde classifier.

3.9 Fault Tolerance

A second scenario where reconfiguration may be useful is for fault tolerance. We could run the larger single bit width design and if an area of the device becomes damaged over the course of time, automatically switch to a smaller design. We can easily check

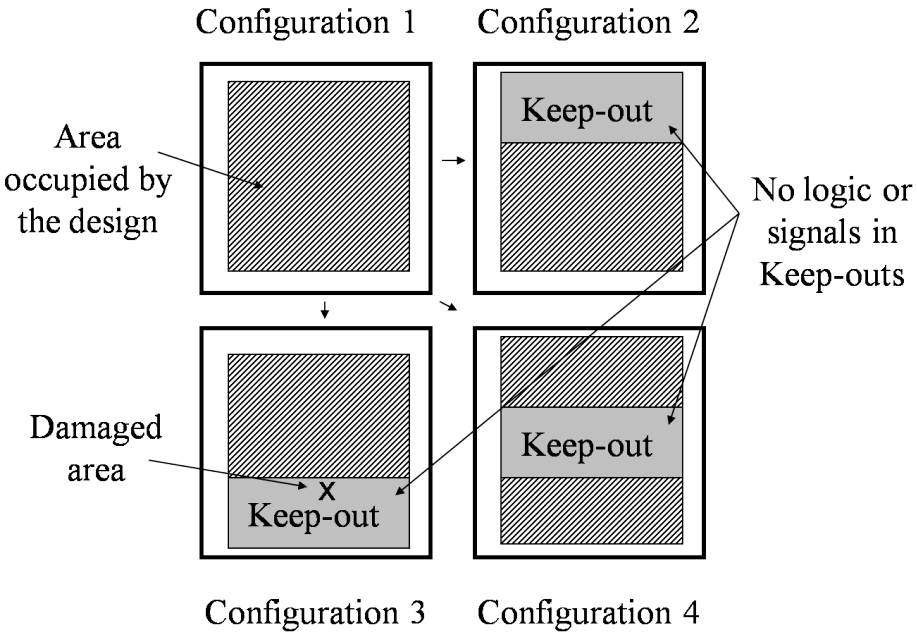


Figure 3.5: Fault Tolerance

the results of our MLP against a reference signal to see if a fault has occurred. If so, we simply reconfigure the device to a smaller design. For instance we could have 4 of the same design stored in memory with each design layout having employing a unique keep-out area as shown in figure 3.5.

3.10 Low Power Dissipation

A third scenario where reconfiguration may be useful is when we wish to switch to a configuration that dissipates less power while maintaining the sampling rate. In this case we choose a Pareto point from Figure 3.4 that uses fewer slices and subsequently wastes less power.

Chapter 4

A Network of Adaptive Classifiers

4.1 Introduction

This chapter focuses on extending the concept of the adaptive classifier, introduced in the previous chapter, to operation in a distributed computing network with decentralized control. In addition, *One versus the-Rest* binary classification is used as opposed to the *One versus One* approach detailed in the previous chapter. The main task of the network is to track and localize chemical clouds over the area covered by the nodes. However, this is only one application and it is believed that the concepts presented here are applicable to other tracking and localization problems using different sensor modalities.

4.2 Network Architecture

The network of adaptive classifiers is a spatial grid consisting of 256 nodes arranged in a 16 x 16 node lattice, but in general can be any dimension. Each node is assigned an address, using the numbers 1 through 256, and nodes are ordered sequentially. Therefore, row number 1 of the network contains nodes with addresses 1 through 16 in sequence, while row number 2 nodes are ordered 17 through 32 etc as shown in Table 4.1.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128
129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144
145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176
177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192
193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208
209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224
225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256

Table 4.1: a network of 256 nodes with address assignments

Nodes are arranged such that each node has four neighbor nodes. These are the four adjacent nodes located on top and bottom and to the left and right, forming what is known as a von Neumann neighborhood. Other neighborhood arrangements can be

found in the literature including the Moore neighborhood consisting of a center node and 8 neighbors, as well as hexagonal neighborhood of nodes having 6 neighbors. The network of adaptive classifiers presented here can be easily adapted to use a Moore neighborhood. However, this will be left for future study and only results of the von Neumann arrangement of nodes will be discussed herein.

The network is populated with adaptive classifiers which are One-versus-the-Rest binary classifiers trained to detect chemical species and individual chemicals belonging to specific species. The classification output may be viewed as a binary decision (or classification) tree where the root, or starting node, is a classifier trained to detect any chemical gas from all other phenomena, a Chemical-versus-the-Rest classifier, and the ending node would be the classified chemical. An example of an individual chemical classifier is a hexanal-versus-the-Rest classifier where hexanal is the chemical identified, and the-Rest refers to the remaining chemicals belonging to the same species; in this case the species in an Aldehyde. Figure 4.2 depicts the binary decision (classification) tree for this example. Figure 4.1 shows the chemical species and individual chemicals included in the simulations.

The classifiers may be arranged in various ways which referred to here as structures. The majority of the experiments of the next chapter use the diagonal structure shown at the end of chapter 5 where the classifiers, numbered 1 through 5, are loaded into the network sequentially by node address, and form diagonal lines within the network. These are the 5 chemical species classifiers which also contain 4 individual chemical classifiers, all at varying bit precisions. It is important to note that only the chemical species classifiers are initially loaded into the reconfigurable logic of the

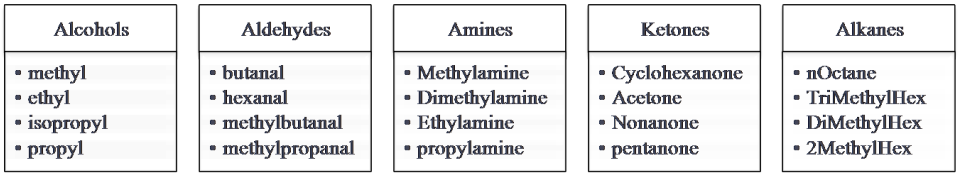


Figure 4.1: 5 Chemical Species/20 Chemicals

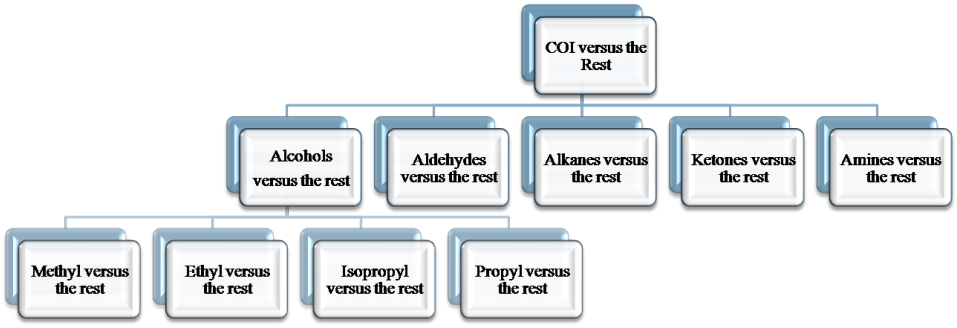


Figure 4.2: Binary Classification

Player 1		Player Two	
		Cooperate	Defect
	Cooperate	R = 3, R =3	S = 0, T = 5
	Defect	T = 5, T = 0	P = 1, P = 1

1. If both players cooperate they each receive 3 points - **reward**
2. If both defect they each receive 1 point – **punishment**
3. If one defects while the other cooperates:
   defector receives 5 points – **temptation**
   cooperator receive 0 points – **suckers payoff**

Figure 4.3: Prisoner's Dilemma

FPGA. Switching the reconfigurable logic to other configurations is based on the decision rules which will be described next.

4.3 The Prisoner's Dilemma

The prisoner's dilemma is a game popularized by game theorists and used by evolutionary biologists, economists and social and political scientists to explore areas as diverse as cellular mechanisms of cancer cells, to the spread of culture and the alignment of nations. Two players meet and each has a choice of cooperating or defecting. If both cooperate then they both receive a reward, R. If they both defect, they receive a punishment, P. If one defects while the other cooperates, then the defector gets a temptation, T, and the cooperator gets a suckers payoff, S, which is usually zero. These are called the payoffs and they are arranged with assigned values in a payoff matrix as shown in Figure 4.3.

In order for the game to be a prisoner's dilemma the following two requirements must be met: $T > R > P > S$ and $2R > T + S$. Therefore the payoffs that each player receives don't necessarily need to be the same. They can be different, provided they meet the above two requirements. The dilemma is that it pays to defect if the other player cooperates, but if both defect, they both do worse than if they had both cooperated. The game is more interesting when played over several iterations and becomes the iterative prisoner's dilemma. It opens up the possibility to employ a strategy to try to maximize ones score over time. The prisoner's dilemma will mostly be used in this way and the strategies that will be explored in this study are ALLC, ALLD, TFT, GTFT, ALTDC and RANDD.

1. ALLC - Always cooperate no matter what done on the previous move(s).

2. ALLD - Always defect no matter what done on the previous move(s).

3. TFT (Tit-For-Tat) - Always cooperate first then do what the other player did previously.

4. GTFT (Generous Tit-For-Tat) - Always cooperate first then cooperate if the other player cooperated on the previous turn or if they previously defected then defect with probability 2/3.

5. ALTDC - Alternate defection and cooperation.

6. RANDD - Randomly defect with probability p = 0.1.

4.4 An Abundance of Classifiers, Growth Rate and Reconfiguration Frequency

In order to assess the effectiveness of classification, tracking and localization a metric is required. The metric used in this study is the percent of correct classifiers under the target chemical cloud, averaged over the duration of the event. This is termed the *abundance* of classifiers and denoted by, A. If A is high, then classification is deemed successful, and conversely, unsuccessful, if it is low. At each sensing opportunity, the total number of correct chemical classifiers that are at the same node locations as the chemical cloud pattern are calculated. Next, the percentage of correct classifiers, in relation to the total number in the cloud pattern is calculated, so as to normalize the values, when comparing against cloud patterns of different sizes. The percentages are then averaged over all sensing opportunities to form the abundance number, A.

Abundance alone does not tell the complete story. With respect to the present task, it is best to form an abundance of classifiers early in the event timeline rather than at the end and for this reason the second metric used is the growth rate of correct classifiers, G. The growth rate is calculated by taking the ratio of the number of correct classifiers at rounds r_{max} over r_1 of the iterative prisoners' dilemma for each sensing opportunity. Therefore when $G > 1$ the population is growing between sensing opportunities, while$G < 1$ implies that the population is dwindling. When $G = 1$, the population remains steady. The figure below shows the growth rate exhibited when playing the strategy ALLD for pattern No. 2, using two sensing cycles and four rounds of prisoner's dilemma per Sensing opportunity (cycle). In this

case the highest rate of growth does appear early enough for tracking and localization to be successful. In this case the abundance number $A = 85.2\%$

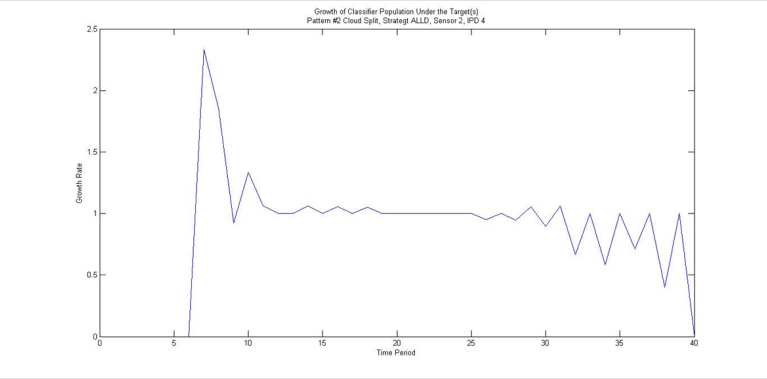


Figure 4.4: Growth Rate

The final metric used in the study is the reconfiguration frequency, F. Each node has an associated value for F. Some strategies may produce higher values of F that others. Higher values of F translate into increased numbers of reconfigurations and communication which then requires more power to operate the network. Given a choice between two strategies with similar A and G, but one with high F and the other with low F, the strategy with low F would be selected.

4.5 Network Parameters

The success or failure of classification, tracking and localization is directly dependent upon several network parameters, including number of sensing opportunities and number of rounds of prisoner's dilemma per sensing opportunity. In addition, the distance between the nodes and/or the bit rate of communication must be matched to the aforementioned parameters. Structure also plays a role, however as stated

previously the focus is on diagonal-structure.

Today, high-speed gas chromatography and Ion Mobility spectrometers can capture and classify a chemical's unique signature in the hundreds of milliseconds. The length of time required for chemical ions to be present in the sensing drift-tube of an IMS device is only 10-25 msec. The adaptive classifier in reconfigurable logic runs most of computing elements in parallel leading to fast execution of the MLP, on the order of milliseconds. Using this information, a sensing and classification time of 1000 ms is used.

In addition, each round of the prisoner's dilemma requires execution of a state machine running within the microcontroller and communication with neighbor nodes. Communication associated with exchanging decisions, strategies and classifier weights is on the order of 600 bytes per round of prisoner's dilemma for 12-bit and 10-bit precision, while 8-bit precision classifiers require only 300 bytes be communicated. Table 4.2 shows achievable communication times under varying precisions using bit rates of 4800 bps, 9600 bps and 19200. The execution time is used to establish a minimum distance between the nodes. Specifically, it is used to match the execution time to the current wind speeds since it is the wind speed that dictates the speed of the chemical cloud. For purposes of this of study, wind speed has been grouped into three categories; fast, medium and slow. Fast clouds travel at 10 m/s, while medium and slow moving clouds travel at 5 m/s and 2.5 m/s, respectively.

As an example, if the target cloud is moving 10 m/s and the bit rate communication is 4800 bps, the nodes would be spaced at least 10m/s x 2 second intervals, or 20 meters apart in order to sense once and play one round of prisoner's dilemma. Multi-

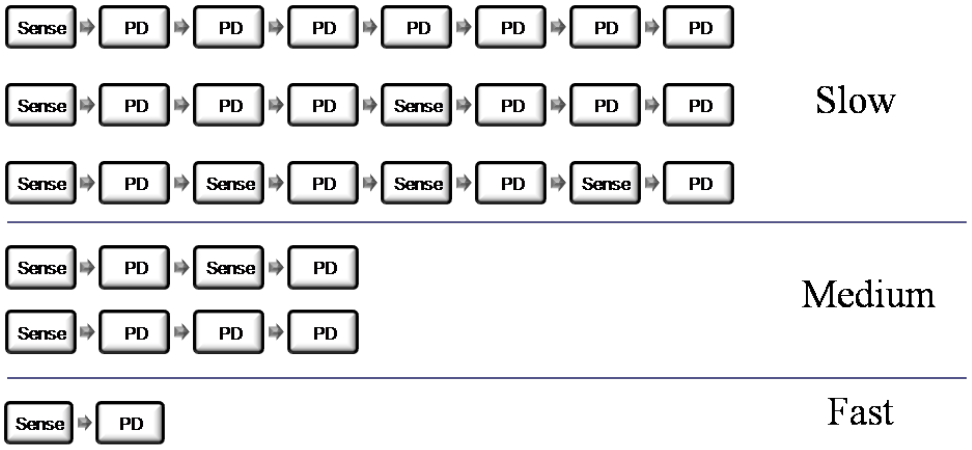


Figure 4.5: Task schedule options at 20 meter node distance

ple rounds are often required to be played and most of the experiments of chapter 5 include at least two rounds per sensing opportunity. This requires that the nodes be placed a minimum distance of 40 meters apart for the scenario given above. At this distance medium and slower moving clouds are given more sensing and game playing opportunities, 2X and 4X more, respectively.

Table 4.2: Exemplar Communication Rates and Times

Bit Rate of Communication	12 & 10 bit Precision	8-bit Precision
4800 bps	1000 ms	500 ms
9600 bps	500 ms	250 ms
19200 bps	250 ms	125 ms

4.6 Networked Adaptive Classifiers

The adaptive classifiers presented in the previous chapter contain one species and 4 chemical classifiers each with 3 levels of precision that are optimized to reduce area/power consumption. Operating them in a network requires that three additional configurations be loaded in the configuration-space of the FPGA. These are full precision versions of the MLP in 12, 10 and 8-bits. These additional configurations allow for execution of the non-native classifiers obtained from dissimilar classifiers through the communication channel as a result of losing a round of the prisoner's dilemma.

The weights and inputs of the optimized classifiers are tuned to various bit lengths and it would be required that the weights as well as their associated bit length be transmitted after every round of game play where at least one node lost. This doubles the number of bit transmitted per round. Even if this extra information were sent each FPGA would need to have the correct arrangement of multipliers and adders which translates into extra configurations. In fact it would be necessary for each classifier to have the configurations of every other classifier for a total of 75 configurations. Each configuration is approximately 6 M-bits in size and having 75 configurations would require a 64 Mbytes of memory whereas 18 configurations can be stored in a much smaller 128 M-bit chip.

Configuration is performed in two general ways. One way is for the reconfigurable logic of the device to be reconfigured from an native classifier to a generic bitwidth classifier. The second way is to reconfigure only the MLP by using weights transmitted from other classif

4.7 Decentralized Control of the Network

While many networks operate under a central authority the evolutionary approach described herein is inherently decentralized. The network is designed to in operate in one of two ways. The first is to employ a single strategy for all nodes to use when playing the prisoner's dilemma. The second is for all nodes to start with the same initial strategy, and then switch to a different one upon sensing a chemical. Both are illustrated using the following example. Tables 4.3 and 4.4 below show a 5 x 5 grid network of adaptive classifiers. The network starts with species classifiers A, B, C, D and E and strategy ALLC, all cooperate. Let's suppose that a chemical gas cloud of species C enters the network at nodes (3, 1) and (4, 1) shown as shaded in the tables. During the next sensing opportunity node (3, 1) detects the chemical species.

Table 4.3: Mode 1 Operation - Single Strategy

	1	2	3	4	5			1	2	3	4	5
1	A	B	C	D	E		1	*C*	*C*	*C*	*C*	*C*
2	B	C	D	E	A		2	*C*	*C*	*C*	*C*	*C*
3	**C**	D	E	A	B		3	*C*	*C*	*C*	*C*	*C*
4	D	E	A	B	C		4	*C*	*C*	*C*	*C*	*C*
5	E	A	B	C	D		5	*C*	*C*	*C*	*C*	*C*

Table 4.4: Mode 2 Operation - Two Strategies

	1	2	3	4	5			1	2	3	4	5
1	A	B	C	D	E		1	*C*	*C*	*C*	*C*	*C*
2	B	C	D	E	A		2	*C*	*C*	*C*	*C*	*C*
3	**C**	D	E	A	B		3	*D*	*C*	*C*	*C*	*C*
4	D	E	A	B	C		4	*C*	*C*	*C*	*C*	*C*
5	E	A	B	C	D		5	*C*	*C*	*C*	*C*	*C*

In operating mode 1 the strategy of node (3, 1) remains the same, while operating in mode 2, the strategy changes to ALLD, always defect. In addition the payoffs for operating mode 1 are $T = 6 > R = 4 > P = 2 > S = 0$ for node (3, 1) and $T = 5 > R = 3 > P = 1 > S = 0$, for all others. In operating mode 2 all nodes including node (3, 1) receive the same payoffs of $T = 5 > R = 3 > P = 1 > S = 0$. Table 4.5 shows the payoffs for each node after one round of the prisoner's dilemma. In operating mode 1, the positively sensing node (3, 1) receives cumulative payoff of 16, because it cooperates once each with its' four neighbors and receive 4 points each time. Its' neighbors have a cumulative payoff of only 12 each. Since they do not sense the chemical they do not receive the extra payoff points allotted to node (3, 1). Since nodes (2, 1), (3, 2) and (4, 1) are all neighbors of node (3, 1) and it has a better payoff than any of their neighbors including themselves, they reconfigure classifiers. Specifically, they switch to one of node (3, 1)'s individual (analyte) chemical classifiers. The choice of which one is determined randomly by node (3, 1) each one given probability 0.25.

Table 4.5: After First Detection

	1	2	3	4	5			1	2	3	4	5
1	12	12	12	12	12		1	A	B	C	D	E
2	12	12	12	12	12		2	C4	C	D	E	A
3	16	12	12	12	12		3	C	C1	E	A	C3
4	12	12	12	12	12		4	C2	E	A	B	C
5	12	12	12	12	12		5	E	A	B	C	D

Let's suppose the chemical cloud contains chemical C2. Table 4.6 below shows the payoffs after a second round of prisoner's dilemma is played prior to the next sensing

opportunity but using the data obtained from the prior sensing opportunity. If this is not done then activity ceases and it makes no sense to play further rounds under mode 1. Also here is only a small computational cost to re-compute an output for the new classifiers. With this consideration, node (4, 1) will positively sense it and receive the increased payoffs.

Table 4.6: Scores and Classifiers

	1	2	3	4	5			1	2	3	4	5
1	12	12	12	12	12		1	A	B	C	D	E
2	12	12	12	12	12		2	C3	C	D	E	A
3	16	12	12	12	12		3	C	C2	E	A	C4
4	16	12	12	12	12		4	C2	C2	A	B	C2
5	12	12	12	12	12		5	C2	A	B	C	D

Node (3, 1) again randomly chooses individual analyte configurations to send to nodes (2, 1), (2, 3) and (3, 5) with probability 0.25 but does not transmit one to node (4, 1) since that node now has an equal score. In addition, this time node (4, 1) transmits its' C2 classifier to nodes (4, 2), (5, 1) and (4, 5). Table 4.7 shows the chemical cloud location at the next time instance and sensing opportunity.

If the cloud contains chemical C2, then the network will correctly classify 2/3 of the cloud and the abundance, $A = (50\% + 66.7\%)/2 = 58.4\%$. If however the cloud contains chemical C3 then only 1/6 of the cloud is correctly classified and $A = 33.3\%$. If the chemical is C1 or C4 then $A = 25\%$. Any classifier that is incorrect after sensing is reset to its' initial species classifier. So, if the cloud contains chemical C2, then nodes (2, 1), (3, 5), and (4, 5) will be reset to their original, B, B and C species classifiers, respectively.

Table 4.7: Classifier Changes

	1	2	3	4	5			1	2	3	4	5
1	A	B	C	D	E		1	A	B	C	D	E
2	C3	C	D	E	A		2	B	C	D	E	A
3	C	C2	E	A	C4		3	C	C2	E	A	B
4	C2	C2	A	B	C2		4	C2	C2	A	B	C
5	C2	A	B	C	D		5	C2	A	B	C	D

Another round of prisoner's dilemma is played and the scores are shown below along with any new updates to the classifiers. It is assumed here that node (3, 1) once again chose the incorrect classifier, C4 to transmit to node (2, 1). If it had chosen C2 instead, then there would be 5/6 correct classifiers under the cloud. Also, if a rule was employed to over-rule the previous tie between parent and child classifier, then node (4, 1) could have reconfigured node (3, 1) with classifier C2, further improving A. Also, the growth rate, $G = 1.5$ under these considerations and $G = 1$ (no growth) without them. Note, in the previous time instance G is undefined because the ratio was 1/0. Reconfiguration frequency $F = 22$ thus far, without the two considerations of correct selection at node (2, 1) and over-ruling ties between parent and child classifiers.

Table 4.8: Updated scores and classifiers

	1	2	3	4	5			1	2	3	4	5
1	12	12	12	12	12		1	C2	B	C	D	E
2	12	12	12	12	12		2	C4	C2	D	E	A
3	16	16	12	12	12		3	C	C2	C2	A	C1
4	16	16	12	12	12		4	C2	C2	C2	B	C2
5	16	12	12	12	12		5	C2	C2	B	C	C2

The mode 2 operation relies on selection of a different strategy upon positive

classification of a species or individual, rather than an increase in payoffs and is explained here.

Using the same cloud pattern at the first time instance node (3, 1) again positively classifies chemical species C. However this time node (3, 1) changes its' strategy from ALLC to ALLD ensuring that it's next decision will be to defect, as indicated in table 4.9. The payoffs after one round of prisoners' dilemma are also given below.

Table 4.9: Mode 2 score, classifiers and decisions

	1	2	3	4	5			1	2	3	4	5			1	2	3	4	5
1	12	12	12	12	12		1	A	B	C	D	E		1	*C*	*C*	*C*	*C*	*C*
2	9	12	12	12	12		2	C1	C	D	E	A		2	*D*	*C*	*C*	*C*	*C*
3	20	9	12	12	9		3	C	C2	E	A	C4		3	*D*	*D*	*C*	*C*	*D*
4	9	12	12	12	12		4	C3	E	A	B	C		4	*D*	*C*	*C*	*C*	*C*
5	12	12	12	12	12		5	E	A	B	C	D		5	*C*	*C*	*C*	*C*	*C*

Node (2, 1), (3, 3), (3, 5) and (4, 1) each receive one of node (3, 1)'s individual chemical classifiers with equal probability, and C1, C2, C4 and C3, respectively. A second round of prisoner's dilemma is played but this time without using the sensor data stored in memory.

Table 4.10: After second round of PD

	1	2	3	4	5			1	2	3	4	5
1	9	12	12	12	12		1	C1	B	C	D	E
2	16	6	12	12	6		2	C1	C1	D	E	C4
3	4	16	9	9	16		3	C2	C2	C2	C4	C4
4	16	6	12	12	6		4	C3	A	A	B	B
5	9	12	12	12	12		5	C3	A	B	C	D

An examination of the ensuing node reconfigurations, reveal that node (1, 1) changes to C1. Node (2, 2) is reconfigured to C1 or C2 with equal probability. Node

(2, 5) can change to C4 or C1. Node (3, 1), the first node to defect, is reconfigured to C1, C2, C3 or C4 with equal probability. Node (3, 3) and (3, 4) change to C2 or C4, respectively. Node (4, 2) is reconfigured to C2 or C3, while node (4, 5) is reconfigured into either C3 or C4. Node (5, 1) is reconfigured to classifier C3. All other nodes keep their configurations for the next sensing opportunity. If on the next sensing opportunity, a node does not identify the chemical as one belonging to its currently loaded configuration, it resets to its' initial configuration and initial strategy, ALLC, resulting in the figures below.

Table 4.11: After Reset

	1	2	3	4	5			1	2	3	4	5
1	A	B	C	D	E		1	*C*	*C*	*C*	*C*	*C*
2	B	C	D	E	A		2	*C*	*C*	*C*	*C*	*C*
3	C2	D	E	A	B		3	*D*	*C*	*C*	*C*	*C*
4	D	E	A	B	C		4	*C*	*C*	*C*	*C*	*C*
5	E	A	B	C	D		5	*C*	*C*	*C*	*C*	*C*

After the next sensing opportunity and round of prisoners' dilemma, classifier C2 at node (3, 1) propagates to nodes (2, 1), (3, 2), (3, 5) and (4, 1). On round two, these nodes will accumulate a score of 16, scoring higher than all other nodes, including node (3, 1) which now has the lowest score of all nodes with a score of 4. However, node (3, 1) does not reconfigure because it already has C2 loaded and is already playing strategy ALLD. The nodes that do reconfigure to classifier C2 and change to ALLD are nodes (1, 1), (2, 2), (2, 5), (3, 3), (3, 4), (4, 2), (4, 5) and (5, 1). Now all classifiers under the cloud are C2 resulting in $A = (0/2+1/6+6/6)\ /2 =$ 38.% as long as the cloud remains in the current position. Even if move to right by

one node there will still be 4 nodes under the cloud, and $A = (0/2+1/6+4/6)\ /2 = 27.8\%$. Even though the abundance seems low, the growth rate $G = 6$, as the number of classifiers under the cloud increased from 1 to 6, which is very good. The downside is that the reconfiguration frequency, $F = 35$.

Table 4.12: Cloud Moves - updated scores/classifiers

	1	2	3	4	5			1	2	3	4	5
1	A	B	C	D	E		1	9	12	12	12	12
2	C2	C	D	E	A		2	16	6	12	12	6
3	C2	C2	E	A	C2		3	4	16	9	9	16
4	C2	E	A	B	C		4	16	6	12	12	6
5	E	A	B	C	D		5	9	12	12	12	12

Table 4.13: Updated decisions and classifiers

	1	2	3	4	5			1	2	3	4	5
1	*C*	*C*	*C*	*C*	*C*		1	C2	B	C	D	E
2	*D*	*C*	*C*	*C*	*C*		2	C2	C2	D	E	C2
3	*D*	*D*	*C*	*C*	*D*		3	C2	C2	C2	C2	C2
4	*D*	*C*	*C*	*C*	*C*		4	C2	C2	A	B	C2
5	*C*	*C*	*C*	*C*	*C*		5	C2	A	B	C	D

Chapter 5 details computer simulations used to assess the effectiveness of both of the operating modes illustrated above but under varying parameters.

Chapter 5

Methods, Experiments and Discussion of Results

5.1 Description of Experiments

In order to test the validity of the approach described in the previous chapters a series of experiments were conducted. Five basic test patterns were presented as input to a computer simulated network of adaptive classifiers. The 5 basic test patterns represent different chemical clouds scenarios. Each test pattern was applied under different network parameters. These parameters represent different ways to run the network and include; number of sensing opportunities, number of iterations of the prisoner's dilemma run per sensing opportunity, different strategies, and classifier network structure. In addition, ion mobility spectra of different volatile organic compounds were used to test the networks classification performance.

5.2 Test Patterns

The following describes in detail the five test patterns used in the experiments. Table 5.1 and the Tables at the end of this chapter, depict each pattern in relation to the 256 node network of classifiers. Each of the numbers 1 through 256 in the figures is the address of a node and also indicates the nodes position in the network. The area over the network that the chemical cloud pattern covers is represented by the shaded nodes. For example, in the case of test pattern number 1, the chemical cloud is shown to extend over the twenty nodes with address numbers 89, 90, 104, 105, 106, 119 through 123, 135 through 139, 151 through 154 and 169. Each pattern moves in a particular direction and manner. In the case of pattern number 1, it moves from West to East, entering and first appearing to the network at nodes 113 and 129 and exiting and last appearing at nodes 128, 144 and 160. The remaining test patterns are depicted in a similar manner as test pattern number 1. However, they exhibit very different motion. Test pattern number 2 is derived from test pattern number 1 and contains the same arrangement of 20 nodes. It is different in that it attempts to simulate what may happen when a moving cloud reaches an obstruction, such a building. This pattern separates into two diverging clouds after blowing into a simulated building located at the nodes with addresses 121 and 137, midway through the network. Test pattern number 3 also contains the same pattern as the separating cloud of pattern number 2; however it contains a small cloud of a different chemical of size 8 nodes moving parallel and in sync with the larger one. After the large cloud separates, the diverging lower half merges with the smaller cloud at nodes in the lower

right corner of the network. While the clouds are merging the sensor senses either of the two chemicals with probability 0.5 therefore the assumption is that the chemicals are not sufficiently mixed to create new spectra but either can still be classifier by their individual ones.

Table 5.1: Cloud Pattern No. 1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128
129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144
145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176
177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192
193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208
209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224
225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256

Test pattern number 4 is quite different in that it models an expanding chemical plume. This type of pattern is typically the result of some type of accident like a train derailment where the train was transporting chemical tanker cars, a pipeline leak or an unintended release at a chemical processing facility. The plume begins at node 61 and over time drifts down and to the left finally stopping at nodes 197 and 213 expanding and contracting slightly. The plume covers 46 nodes at its greatest extent. Finally, test pattern number 5 simulates one large cloud moving from North to South followed immediately by a smaller cloud of a different chemical. The motion attempts to simulate a chemical attack whereby the attacker attempts to trick the

network into reacting to the first and misclassify, or miss completely, the second one, which would presumably be a more lethal substance.

5.3 Group 1 versus Group 2 Experiments

Experiments are grouped by the parameter settings applied during the simulation run. In both groups sensing opportunities are varied between 1, 2 and 4, also the number of iterations of the prisoners' dilemma was usually run at 2 or 3 iterations and occasionally more. Classifiers were arranged in 1 of three different ways. All of the experiments presented use the diagonal structure shown in Appendix A, where each species classifier is assigned sequentially to sequential node addresses 1 through 256. A number is assigned to each of the 25 classifier types. Classifiers were assigned numbers 1 through 5 for the species and 11, 12, 13 and 14 for species 1, etc. The two other classier arrangements were used periodically to test if node neighborhood influenced the tracking and localization performance. Results from these trials are not presented herein. The first of these is a random assignment of classifiers while the second aligns the classifiers in columns while also eliminating one class entirely.

Group 1 experiments are characterized by the fact that they all use a single strategy when playing the prisoners' dilemma. All nodes of the network play only one of the following strategies; ALLC, TFT, ALLD, ALTDC and RANDD and all nodes play the same strategy. No switching of strategies is permitted. The driving force behind the node reconfigurations is an imbalance in payoffs. Nodes that correctly classify a chemical species or individual chemical receive an extra payoff point. In

Group 2 experiments nodes play one of two strategies, including ALLC, ALLD, TFT, GTFT, ALTDC, and RANDD with each strategy having equal payoffs. In this way it is possible to test one strategy versus another.

5.3.1 Group 1, Experiment No. 1

These simulations use pattern number 2 and a single strategy to reconfigure the nodes.

Positive Sensing Node: $T = 57 > R = 27 > P = 7 > S = 0$

Negative Sensing Node: $T = 56 > R = 26 > P = 6 > S = 0$

The chemical spectra used: Methyl Alcohol (Alcohol Species)

Table 5.2: Group 1 Experiment No. 1 Results for TFT ALLC GTFT

TFT/ALLC/GTFT - Patten No. 2

Two Rounds PD	Run#					A
Sensor Cycles	1	2	3	4	5	Average
1	14.00%	33.60%	35.00%	29.40%	44.80%	31.30%
2	71.30%	83.10%	83.10%	82.40%	51.20%	74.20%
4	81.80%	85.00%	71.10%	87.00%	70.40%	79.10%
8	80.60%	87.60%	94.60%	94.70%	88.70%	89.30%
16	95.40%	88.60%	81.00%	95.30%	95.40%	91.10%

The results indicate that the ALTDC strategy is slightly better at classifying the chemical cloud than TFT/ALLC/GTFT when sensing opportunities are 1, 2, 4 or 8. TFT/ALLC/GTFT is slightly better when there are 16 sensing opportunities. ALLD is better than either ALTDC or TFT/ALLC/GTFT when sensing opportunities are 1 or 4 but otherwise, considerably worse. RANDD was unacceptably bad with the

Table 5.3: Group 1 Experiment No. 1 Results for ALTDC

ALTDC

Two Rounds PD	Run No.					A
Sensor Cycles	1	2	3	4	5	Average
1	40.60%	39.20%	20.60%	41.60%	22.70%	32.90%
2	80.00%	79.40%	76.80%	81.40%	77.00%	78.90%
4	88.80%	86.70%	59.90%	88.30%	86.60%	82.10%
8	94.30%	87.40%	94.60%	87.40%	87.80%	90.30%
16	89.20%	95.40%	95.40%	95.50%	74.20%	89.90%

Table 5.4: Group 1 Experiment No. 1 Results for ALLD

ALLD

Two Rounds PD	Run No.					A
Sensor Cycles	1	2	3	4	5	Average
1	46.90%	34.30%	35.30%	25.90%	38.80%	36.20%
2	79.60%	6.30%	81.40%	83.10%	71.80%	64.40%
4	78.90%	86.80%	86.50%	85.80%	89.70%	85.50%
8	87.70%	94.60%	74.30%	75.40%	87.80%	84.00%
16	95.40%	81.50%	80.80%	89.60%	55.80%	80.60%

Table 5.5: Group 1 Experiment No. 1 Results for RANDD

RANDD

Two Rounds/SC	Run No.					A
Sensor Cycles	1	2	3	4	5	Average
4	3.50%	22.00%	5.50%	8.30%	12.10%	10.30%

very low value of A at 10.3%. For very fast moving clouds the data shows it would be best to run the ALLD strategy. If however the clouds are moving at a moderate pace either ALTDC or ALLD should be used. For slow moving clouds a strategy of TFT/ALLC/GTFT or ALTDC should be implemented.

5.3.2 Group 1, Experiment No. 2

These simulations use pattern number 3 where two gas clouds move at the same pace parallel to each other for a time. The larger one separates into two clouds after striking a simulated building. Results for the two best performing strategies are presented in the tables below. Payoffs used are as follows:

Positive Sensing Node: $T = 57 > R = 27 > P = 7 > S = 0$ Negative Sensing Node:

$$T = 56 > R = 26 > P = 6 > S = 0$$

The chemical spectra used: Large Cloud: Methyl Alcohol (Alcohol Species) Small Cloud: nOctane (Alkane Species)

Table 5.6: Group 1, Experiment No. 2 Results using TFT

		TFT Alone				
Chemical	Sensing Opportunities	Rounds of Prisoner's Dilemma	Average Abundance	Standard Deviation	Abundance Max.	Abundance Min.
methyl alcohol	1	3	29.2%	13.9%	51.3%	0.4%
nOctane	1	3	12.2%	21.1%	63.3%	0.0%
methyl alcohol	2	3	56.0%	14.5%	80.1%	19.5%
nOctane	2	3	32.8%	42.8%	95.9%	0.0%
methyl alcohol	4	3	61.6%	22.7%	84.9%	0.4%
nOctane	4	3	55.0%	45.7%	96.8%	0.0%

Table 5.7: Group 1, Experiment No. 2 Results using ALLC

		ALLC Alone				
Chemical	Sensing Opportunities	Rounds of Prisoner's Dilemma	Average Abundance	Standard Deviation	Abundance Max.	Abundance Min.
methyl alcohol	1	3	30.4%	10.8%	47.8%	0.0%
nOctane	1	3	24.4%	25.4%	62.5%	0.0%
methyl alcohol	2	3	59.7%	18.4%	80.0%	0.0%
nOctane	2	3	29.2%	41.4%	95.4%	26.8%
methyl alcohol	4	3	65.0%	16.0%	87.6%	0.0%
nOctane	4	3	54.3%	46.2%	96.4%	1.8%

Results for TFT and ALLC are provided in the tables and indicate that at best it can correctly classify between 55% and 65% of the two clouds. ALLC had slightly better results than TFT.

5.3.3 Group 2, Experiment No. 1

These simulations use pattern number 1 where a single chemical clouds moves from West to East. The chemical spectrum used was methyl alcohol (alcohol species). In this experiment the element of randomness was reduced by allowing each species classifier to have only one individual classifier instead of the usual four. This also served to help establish the maximum value that could be expected for A.

$$\text{Payoffs: } T = 1.67 > R = 1 > P = 0.67 > S = 0$$

This experiment reveals that under the best circumstances an abundance averaging in the low to middle 90 percentile can be expected. This infers that very good classification can be expected if values of A range in the middle to high 80 percentile.

Table 5.8: Group 2, Experiment No. 1A Results using ALLD vs. ALLC

Cloud Pattern No. 1

ALLD versus ALLC

Two Rounds PD	Run No.					A
Sensor Cycles	1	2	3	4	5	Average
1	91.80%	90.30%	91.50%	92.20%	91.50%	91.50%
2	86.80%	95.80%	95.10%	93.10%	92.30%	92.60%
4	82.70%	96.80%	96.50%	96.70%	96.70%	93.90%

Table 5.9: Group 2, Experiment No. 1A Results using ALLD vs. TFT

Cloud Pattern No. 1

ALLD versus TFT

Two Rounds PD	Run No.					A
Sensor Cycles	1	2	3	4	5	Average
1	91.50%	91.80%	90.60%	92.20%	92.20%	91.70%
2	95.80%	95.80%	92.50%	95.90%	87.10%	93.40%
4	96.30%	96.80%	96.70%	96.60%	96.80%	96.60%

Next ALLD versus ALLC and ALLD versus TFT are compared. Here the species classifiers have a one in four chance of forwarding the correct individual chemical classifier on the first attempt. This represents the normal operation of the network.

Table 5.10: Group 2, Experiment No. 1B Results

Cloud Pattern No. 1

ALLD versus ALLC

Two Rounds PD	Run No.						
Sensor Cycles	1	2	3	4	5	A	F
1	79.90%	88.70%	92.20%	63.90%	89.30%	82.80%	1273
2	61.60%	68.20%	92.90%	90.90%	92.30%	81.20%	2495
4	98.00%	56.00%	97.30%	98.00%	93.70%	88.80%	5464
ALLD versus TFT							
1	92.80%	81.80%	89.30%	88.10%	81.20%	86.60%	1331
2	96.40%	87.30%	89.30%	95.30%	73.50%	88.40%	2655
4	93.70%	94.40%	98.00%	89.40%	92.50%	93.60%	5506

Here we see that TFT outperforms ALLC with only a very slight increase in reconfiguration frequency.

5.3.4 Group 2, Experiment No. 2

These simulations use pattern number 2 where a single chemical cloud moves from West to East then strikes a simulated building and separates into two diverging clouds. Here again the species classifiers have a one in four chance of forwarding the correct individual chemical classifier on the first attempt. This represents the normal operation of the network and all experiments that follow will be for the normal operating

mode. To maintain consistency, methyl alcohol is once again the chemical used for the simulation. In addition, the cyber-physical system is described in terms of the network node distances and particular schedule of sensing opportunities and rounds of the prisoner's dilemma. The reader may wish to refer back to figure 4.4 to review the schedule of tasks at 20 meter node distances.

$$\text{Payoffs: } T = 1.67 > R = 1 > P = 0.67 > S = 0$$

Table 5.11: Results for CPS with 40 meter node spacing ALLC vs. ALLD

ALLC versus ALLD

Cloud Velocity	Sensing Opportunities	Rounds of Prisoner's Dilemma	Average Abundance	Standard Deviation	Average Frequency	Standard Deviation
Fast	1	1	28.3%	11.8%	275	66
	1	3	80.7%	19.0%	2575	158
	2	1	68.1%	15.6%	921	148
Medium	1	7	77.8%	16.9%	10293	1582
	2	3	89.2%	10.6%	5295	216
	4	1	79.1%	11.2%	2023	216
Slow	1	15	73.7%	29.7%	39617	6159
	2	7	78.6%	23.0%	21198	3170
	4	3	90.7%	13.9%	10559	781
	8	1	76.0%	11.7%	4032	329

Observation of the results indicates that the optimal number of rounds of the prisoner's dilemma is three. All three categories of cloud velocity responded best when three rounds were played. They achieved scores of 80.7%, 89.7% and 90.7% for fast, medium and slow moving clouds, respectively with only a moderate level of reconfiguration frequency. It is important to note that while the table contains classifier abundance values for an inter-node distance of 40 meters, the same results

may be used for a network with 20 meter node distance. In the case of 20 meters, only one sensing opportunity and one round of prisoner's dilemma could be executed for the fast cloud velocity. For a medium cloud velocity 1 and 3 or 2 and 1 of each could be executed. Lastly, slow moving clouds have all three options of the medium velocity clouds at 40 meter distance.

Table 5.12: Results for CPS with 40 meter node spacing TFT and GTFT

Cloud Velocity	Sensing Opportunities	Rounds of Prisoner's Dilemma	Average Abundance	Standard Deviation	Average Frequency	Standard Deviation
			TFT vs. ALLD			
Fast	1	3	87.4%	6.2%	2880	75
Med	2	3	87.6%	11.6%	5716	242
Slow	4	3	89.3%	14.6%	11540	453
			GTFT vs. ALLD			
Fast	1	3	81.9%	19.7%	2716	352
Med	2	3	81.1%	21.7%	5359	662
Slow	4	3	83.6%	22.7%	10918	1183

Guided by the previous results other strategies were used in simulations and the table above shows results of TFT and GTFT when playing three rounds of prisoner's dilemma. In this case TFT versus ALLD outperforms the previous strategy of ALLC versus ALLD. It did particularly better when classifying the fast chemical cloud. Next, other chemicals were used in simulations with TFT versus ALLD for fast moving clouds. The results, shown below, indicate that TFT versus ALLD is indeed a good strategy set to use as it successfully classified the amine, aldehyde and alkane. However, the alkane (nOctane) was misclassified 5 times out of 25 runs. Further classifier training may be helpful or perhaps introducing a one-versus-one classifier would help (i.e. nOctane versus butanal), the misclassified chemical. This

is left for further study.

Table 5.13: Other Chemicals using TFT versus ALLD and Fast Cloud Velocity

Chemical	Sensing Opportunities	Rounds of Prisoner's Dilemma	Average Abundance	Standard Deviation	Abundance Max.	Abundance Min.
MethylAmine	1	3	83.6%	17.9%	99.0%	24.8%
Butanal	1	3	74.9%	17.0%	87.4%	16.4%
nOctane	1	3	70.9%	40.0%	99.0%	0.0%

5.3.5 Group 2, Experiment No. 3

Experiment No. 3 uses cloud pattern number 3 with clouds moving parallel then merging after the larger cloud strikes a simulated building. First methyl alcohol and nOctane are used then nOctane is switched with hexanal. Simulations are also conducted using different payoffs. The following are used first and results are compiled in table 5.15.

$$\text{Payoffs used: } T = 1.67 > R = 1 > P = 0.67 > S = 0$$

Next, the following payoffs were used: $T = 56 > R = 26 > P = 6 > S = 0$ for results in table 5.17.

Table 5.18 shows results using payoffs: $T = 5 > R = 3 > P = 1 > S = 0$ which are similar to the original payoffs but with a smaller proportional value for the punishment.

Finally, a change is made to the chemical used for the large cloud. It is switched from methyl alcohol to hexanal and the new chemical is simulated using the original payoffs with much improved results. This implies that nOctane and methyl alcohol

Table 5.14: Group 2, Experiment No. 3 Results using ALLD vs. TFT

Cloud Pattern No. 3

Large Chemical Cloud : Methyl Alcohol

ALLD versus TFT

Two Rounds PD	Run No.					A
Sensor Cycles	1	2	3	4	5	Average
1	67.00%	68.30%	64.90%	69.50%	73.50%	68.60%
2	49.10%	0.00%	65.30%	0.20%	45.20%	32.00%
4	18.40%	0.00%	0.00%	51.40%	86.00%	31.20%

Small Chemical Cloud: nOctane

ALLD versus TFT

Two Rounds PD	Run No.					A
Sensor Cycles	1	2	3	4	5	Average
1	79.80%	79.20%	92.40%	0.00%	0.00%	50.30%
2	0.00%	96.60%	72.00%	0.00%	94.60%	52.60%
4	97.20%	0.00%	0.00%	94.20%	0.00%	38.30%

Table 5.15: Group 2, Experiment No. 3 Results using ALLD vs. ALLC

Cloud Pattern No. 3

Large Chemical Cloud : Methyl Alcohol

ALLD versus ALLC

Two Rounds PD	Run No.					A
Sensor Cycles	1	2	3	4	5	Average
1	0.00%	64.40%	71.20%	75.00%	75.50%	57.20%
2	56.80%	0.00%	48.90%	51.80%	0.00%	31.50%
4	0.00%	4.60%	93.00%	49.70%	0.10%	29.50%

Small Chemical Cloud: nOctane

ALLD versus ALLC

Two Rounds PD	Run No.					A
Sensor Cycles	1	2	3	4	5	Average
1	95.00%	94.00%	0.00%	0.00%	58.30%	49.50%
2	0.00%	96.60%	94.70%	94.10%	96.70%	76.40%
4	96.70%	96.80%	0.40%	94.60%	97.00%	77.10%

Table 5.16: Results (above) and Cloud Map (below) for several time instances

	1	2	3	4			1	2	3	4			1	2	3	4
1	1	2	3	4		1	1	2	3	4		1	1	2	3	4
2	2	3	4	5		2	2	3	4	5		2	2	3	4	5
3	3	4	5	1		3	3	4	5	1		3	3	4	5	1
4	4	5	1	2		4	4	5	1	2		4	4	5	1	2
5	11	1	2	3		5	5	1	2	3		5	5	1	2	3
6	1	2	3	4		6	1	2	3	4		6	1	2	3	4
7	2	3	4	5		7	2	3	4	5		7	2	3	4	5
8	3	4	5	1		8	3	4	5	1		8	31	4	5	1
9	4	5	1	2		9	31	5	1	2		9	31	31	1	2
10	5	1	2	3		10	31	31	2	3		10	31	31	2	3
11	31	2	3	4		11	31	2	3	4		11	31	2	3	4
12	31	31	4	5		12	31	31	4	5		12	2	31	4	5
13	31	31	31	1		13	31	31	31	1		13	23	31	31	1
14	31	31	31	2		14	31	31	31	2		14	31	31	31	2
15	31	31	2	3		15	31	31	2	3		15	5	31	2	3
16	31	2	3	4		16	31	2	3	4		16	1	2	3	4

	1	2	3	4			1	2	3	4			1	2	3	4
1	0	0	0	0		1	0	0	0	0		1	0	0	0	0
2	0	0	0	0		2	0	0	0	0		2	0	0	0	0
3	0	0	0	0		3	0	0	0	0		3	0	0	0	0
4	0	0	0	0		4	0	0	0	0		4	0	0	0	0
5	0	0	0	0		5	0	0	0	0		5	0	0	0	0
6	11	0	0	0		6	11	0	0	0		6	11	11	0	0
7	11	0	0	0		7	11	0	0	0		7	11	11	0	0
8	11	11	0	0		8	11	11	0	0		8	11	11	11	0
9	11	11	0	0		9	11	11	0	0		9	11	11	11	0
10	11	0	0	0		10	11	0	0	0		10	11	11	0	0
11	0	0	0	0		11	0	0	0	0		11	11	0	0	0
12	31	0	0	0		12	31	0	0	0		12	0	31	0	0
13	31	31	0	0		13	31	31	0	0		13	31	31	31	0
14	31	31	0	0		14	31	31	0	0		14	31	31	31	0
15	31	0	0	0		15	31	0	0	0		15	0	31	0	0
16	0	0	0	0		16	0	0	0	0		16	0	0	0	0

Table 5.17: Group 2, Experiment No. 3 Results using Different Payoffs (1)

Cloud Pattern No. 3

Large Chemical Cloud : Methyl Alcohol

ALLD versus ALLC

Two Rounds PD	Run No.					A
Sensor Cycles	1	2	3	4	5	Average
1	40.30%	66.20%	70.50%	69.10%	64.20%	62.00%
2	79.50%	0.00%	84.50%	40.90%	48.20%	50.60%

Small Chemical Cloud: nOctane

ALLD versus ALLC

Two Rounds PD	Run No.					A
Sensor Cycles	1	2	3	4	5	Average
1	0.00%	92.50%	14.60%	0.00%	96.60%	40.80%
2	0.00%	30.80%	0.00%	0.00%	0.00%	6.20%

Table 5.18: Group 2, Experiment No. 3 Results using Different payoffs (2)

Large Chemical Cloud : Methyl Alcohol

ALLD versus ALLC

Two Rounds PD			Run No.			A
Sensor Cycles	1	2	3	4	5	Average
1	40.10%	11.90%	62.10%	33.00%	67.80%	43.00%
2	29.70%	0.00%	44.30%	19.80%	56.30%	30.00%

Small Chemical Cloud: nOctane

ALLD versus ALLC

Two Rounds PD			Run No.			A
Sensor Cycles	1	2	3	4	5	Average
1	47.10%	14.90%	0.00%	19.20%	0.00%	16.20%
2	3.30%	24.40%	77.30%	2.00%	0.00%	21.40%

could benefit from more training or the use of some other technique. Payoffs: $T = 1.67 > R = 1 > P = .67 > S = 0$.

5.3.6 Group 2, Experiment No. 4

This experiment simulates a spreading plume the type exhibited by a pipeline leak, train car derailment or leak at a chemical processing facility. Once again TFT versus ALLD outperforms ALLC versus ALLD.

Payoffs: $T = 1.67 > R = 1 > P = .67 > S = 0$

Table 5.19: Group 2, Experiment No. 3 Results using Hexanal

Large Chemical Cloud : Hexanal

ALLD versus ALLC

Two Rounds PD	Run No.					A
Sensor Cycles	1	2	3	4	5	Average
1	88.10%	71.90%	49.10%	67.40%	54.50%	66.20%
2	89.10%	92.30%	89.50%	92.80%	91.90%	91.10%

Small Chemical Cloud: nOctane

ALLD versus ALLC

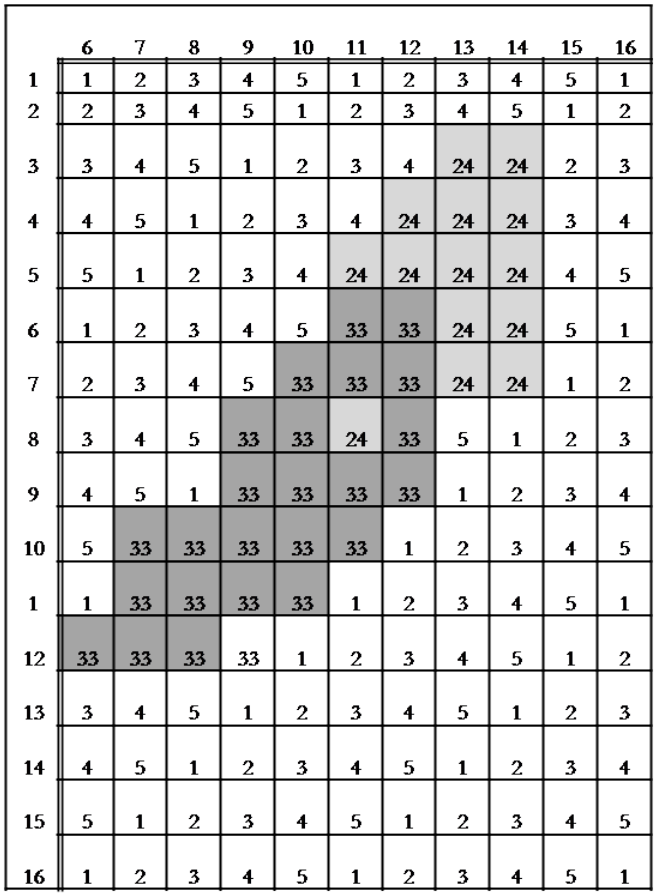
Two Rounds PD	Run No.					A
Sensor Cycles	1	2	3	4	5	Average
1	90.90%	70.00%	95.80%	15.10%	14.30%	57.20%
2	94.60%	69.30%	94.50%	94.20%	0.00%	70.50%

Table 5.20: Group 2, Experiment No. 4 Results using ALLC vs. ALLD

Pattern No. 4, Plume : Hexanal No. 24

ALLD versus ALLC

Two Rounds PD	Run No.					A
Sensor Cycles	1	2	3	4	5	Average
1	89.10%	59.50%	68.30%	94.90%	72.40%	76.80%
2	97.00%	92.30%	95.40%	60.50%	53.10%	79.70%
4	58.80%	82.60%	44.20%	96.50%	38.60%	64.10%



	6	7	8	9	10	11	12	13	14	15	16
1	1	2	3	4	5	1	2	3	4	5	1
2	2	3	4	5	1	2	3	4	5	1	2
3	3	4	5	1	2	3	4	24	24	2	3
4	4	5	1	2	3	4	24	24	24	3	4
5	5	1	2	3	4	24	24	24	24	4	5
6	1	2	3	4	5	33	33	24	24	5	1
7	2	3	4	5	33	33	33	24	24	1	2
8	3	4	5	33	33	24	33	5	1	2	3
9	4	5	1	33	33	33	33	1	2	3	4
10	5	33	33	33	33	33	1	2	3	4	5
1	1	33	33	33	33	1	2	3	4	5	1
12	33	33	33	33	1	2	3	4	5	1	2
13	3	4	5	1	2	3	4	5	1	2	3
14	4	5	1	2	3	4	5	1	2	3	4
15	5	1	2	3	4	5	1	2	3	4	5
16	1	2	3	4	5	1	2	3	4	5	1

Figure 5.1: Results from Run No. 5 Sensor Cycle No. 4 (ALLC)

Table 5.21: Group 2, Experiment No. 4 Results using TFT vs. ALLD

Pattern No. 4, Plume : Hexanal No. 24

ALLD versus TFT

Two Rounds PD			Run#			A
Sensor Cycles	1	2	3	4	5	Average
1	86.70%	74.40%	88.20%	79.70%	87.70%	83.30%
2	63.50%	96.60%	95.70%	96.10%	95.20%	89.40%
4	68.80%	39.80%	32.70%	98.70%	98.70%	67.70%

	6	7	8	9	10	11	12	13	14	15	16
1	1	2	3	4	5	1	2	3	4	5	1
2	2	3	4	5	1	2	3	24	24	1	2
3	3	4	5	1	2	3	24	24	24	24	3
4	4	5	1	2	3	24	24	24	24	3	4
5	5	1	2	3	24	24	24	24	24	4	5
6	1	2	3	4	24	24	24	24	24	5	1
7	2	3	4	24	24	24	24	24	24	24	2
8	3	4	24	24	24	24	24	11	24	2	3
9	4	24	31	24	24	24	24	24	2	3	4
10	24	24	24	24	24	24	24	2	3	4	5
1	24	24	24	24	24	24	2	3	4	5	1
12	24	24	33	24	24	2	3	4	5	1	2
13	24	31	12	24	2	3	4	5	1	2	3
14	4	5	1	2	3	4	5	1	2	3	4
15	5	1	2	3	4	5	1	2	3	4	5
16	1	2	3	4	5	1	2	3	4	5	1

Figure 5.2: Results from Run No. 5 Sensor Cycle No. 4 (TFT)

5.3.7 Group 2, Experiment No. 5

In this experiment one large cloud attempts to mask a second smaller cloud from the network. Several simulations using ALLC versus ALLD and TFT versus ALLD were tried, however the network was less successful than in previous experiments. It was only able to classify the smaller cloud in 1 out of 4 attempts.

5.4 Concluding Remarks

This dissertation presented Networked Adaptive Classification, a decentralized control strategy for a cyber-physical system using techniques developed for game theory. Simulations show that the network can successfully classify, track and localize the ion mobility spectra data for which it was trained. While the network was successful at classifying some chemicals and scenarios better than others further tuning of parameters and training of the MLP's would certainly improve results of this first attempt, even further.

Table 5.22: Diagonal structure for classifiers

1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1
2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2
3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3
4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4
5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1
2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2
3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3
4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4
5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1
2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2
3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3
4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4
5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1

Table 5.23: Cloud Pattern No. 2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
113	114	115	116	117	118	119	120			123	124	125	126	127	128
129	130	131	132	133	134	135	136			139	140	141	142	143	144
145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176
177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192
193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208
209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224
225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256

Table 5.24: Cloud Pattern No. 3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
113	114	115	116	117	118	119	120			123	124	125	126	127	128
129	130	131	132	133	134	135	136			139	140	141	142	143	144
145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176
177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192
193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208
209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224
225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256

Table 5.25: Cloud Pattern No. 4, Plume

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128
129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144
145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176
177	178	179	180	181	182	183	184	185		187	188	189	190	191	192
193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208
209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224
225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256

Table 5.26: Cloud Pattern No. 5

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128
129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144
145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176
177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192
193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208
209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224
225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256

Bibliography

[1] Eiceman, G. A. and Karpas, Z., *Ion Mobility Spectrometry, 2nd Edition*, Taylor Francis Group, CRC Press 2005.

[2] United States National Science Foundation, *Program Solicitation NSF 08-611*, September 2008.

[3] von Neumann, J. and Morganstern, O., *Theory of Games and Economic Behavior*, Princeton University Press, 1944.

[4] Trivers, R. L., The evolution of reciprocal altruism. *Quarterly Review of Biology*, 46, 35-57, 1971.

[5] Maynard Smith, J. and Price, G. R., The logic of animal conflict. *Nature* 246: 15-18, 1973.

[6] Poundstone, W., *Prisoner's Dilemma*, Doubleday, New York, NY, 1992.

[7] Axelrod, R. *The Evolution of Cooperation*. BasicBooks, 1984.

[8] Nowak, M.A. *Evolutionary Dynamics, Exploring the Equations of Life*. The Belknap Press of Harvard University Press. 2006.

[9] Langer, P. Martin, N.A. Hauert, C. Spatial invasion of cooperation. *Journal of Theoretical Biology*, 250, 2008, 634-641.

[10] Axelrod, R. *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*. Princeton University Press, 1997.

[11] Mittal, S and Deb, K. Optimal Strategies of the Iterated Prisoner's Dilemma Problem for Multiple Conflicting Objectives. *IEEE Transactions on Evolutionary Computation.* vol. 13, No. 3, July 2009.

[12] Garcin, F. Manshaei, M.H. Hubaux, J-P. Cooperation in Underwater Sensor Networks.Laboratory of Computer Communications and Application, EPFL, Switzerland.

[13] Hines, P. Balasubramaniam, K. Sanchez, E.C. Cascading failures in power grids. *IEEE Potentials*. 2009.

[14] Brooks, C. Iagnemma, K. Visual Detection of Novel Terrain via Two-Class Classification. *IEEE Transactions on Instrumentation and Measurement*, Vol. 55, No.3, June 2006.

[15] Hill, S and Doucet, A. Adapting Two-Class Support Vector Classification Methods to Many Class Problems. *Proceedings of the 22nd International Conference on Machine Learning*. Bonn, Germany 2005.

[16] Ayech, H.E. and Trabelsi, A. Decomposition Method for Neural Multiclass Classification Problem. *Proceeding of World Academy of Science, Engineering and Technology*. vol. 15, October 2006.

[17] Evan-Zohar Y., Roth, D. A Sequential Model for Multi-Class Classification. {EMNLP 2001

[18] Bermak, A., Belhouari, S.B., Bayesian Learning Using Gaussian Process for Gas Identification.*IEEE Transactions on Instrumentation and Measurement*, vol. 55, No. 3, June 2006.

[19] Shi, M., Bermak, A., Chandrasekaran, S., Amira, A., Brahim-Belhouari, S. A., Committee Machine Gas Identification System Based on Dynamically Reconfigurable FPGA. *IEEE Sensors Journal*. vol. 8, No. 4. April 2008.

[20] Roehl, J.E., Ion Mobility Spectrometry (IMS) - A Chemical Separation Technique using an Electrostatic Field. 89Ch2792-0/89/0000-2190, IEEE 1989.

[21] Kirleis, E. On-Site Trace Chemical Detection, Part 1: Understanding Ion Mobility and Differential Mobility Spectroscopy. *Sensors Magazine*, January 1st, 2008.

[22] Hartman, J., Prouty, W., Moll, A., Hill, H., Gribb, M., Control and Signal Processing for an IMS Sensor. ECE Department, Boise State University, Boise ID, USA.

[23] Bell, S., Nazarov,E., Wang, Y. F., and Eiceman, G. A., Classification of ion mobility spectra by functional group using neural networks. *Analytica Chimica ACTA*, 394(1999) 121-133, Elsevier, May 14, 1999.

[24] Eiceman, G. A., Wang, M., Prasad, S., Schmidt, H., Tadjimukhamedov, F.K., Lavine, B. K., and Mirjankar, N., Pattern recognition analysis of differential mobility spectra with classification by chemical family. *Analytica Chimica ACTA*, 579(2006) 1-10,Elsevier, July 14, 2006.

[25] Gaffer, A. A., Mencer, O., Luk, W., Cheung, P., Unifying Bit-Width Optimization for Fixed-point and Floating-point Designs. *Proceeding of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'04)*. IEEE Computer Society. 2004.

[26] Griewank, A., and Corlis, G. F., Editors. *Automatic Differentiation of Algorithms: Theory, Implementation and Application*. Society for Industrial and Applied Mathematics.

[27] Lee, D., Gaffar, A. A., Mencer, O. and Luk, W., MiniBit: Bit-Width Optimization via Affine Arithmetic. *DAC 2005*, ACM.

[28] Gaffar, A. A., Mencer, O., Luk, W. and Cheung, P., Unifying Bit-Width Optimization for Fixed-point and Floating-point Designs. *Proceeding of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'04)*, IEEE Computer Society, 2004.

[29] Gaffar, A. A., Clarke, J. A. and Constantinides, G. A., PowerBit - Power Aware Arithmetic Bit-Width Optimization. IEEE 2006.

[30] Kum, K., Sung, W., Combined Word-Length Optimization and High-Level Synthesis of Digital Signal Processing Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. vol. 20 No.8, August 2001.

[31] Kanu, A. B., Hill Jr., H. H., Gribb, M. M., and Walters, R. N., A small subsurface ion mobility spectrometer sensor for detecting environmental soil-gas contaminants. *Journal of Environmental Monitoring*, 2007, 9, pp. 51-60. The Royal Society of Chemistry 2007.

[32] Gilberti, M. and Doboli, A. Adaptive Precision Neural Networks for Image Classification. *NASA/ESA Conference on Adaptive Hardware and Systems*, 2008.