

# Applying a Neural Network to Recover Missed RFID Readings

Peter Darcy

Bela Stantic

Abdul Sattar

Institute for Integrated and Intelligent Systems  
Griffith University, Queensland, Australia,  
Email: {P.Darcy, B.Stantic, A.Sattar}@griffith.edu.au

## Abstract

Since the emergence of Radio Frequency Identification technology (RFID), the community has been promised a cost effective and efficient means of identifying and tracking large sums of items with relative ease. Unfortunately, due to the unreliable nature of the passive architecture, the RFID revolution has been reduced to a fraction of its intended audience due to anomalies such as missed readings. Previous work within this field of study have focused on restoring the data at the recording phase which we believe does not allow enough evidence for consecutive missed readings to be corrected. In this study, we propose a methodology of intelligently imputing missing observations through the use of an *Artificial Neural Network* (ANN) in a static environment. Through experimentation, we discover the most effective algorithm to train the network is via genetic training with a high chromosome population. We also establish that the ANN restores a cleaner data set than other intelligent classifier methodologies in the majority of the test cases especially when faced with large amounts of missing data.

*Keywords:* Radio Frequency Identification - RFID, Artificial Neural Network, Data Cleaning.

## 1 Introduction

The alluring promise of an automatic and wireless technology that is cost effective and efficient has been the main appeal for researchers to find a feasible way to implement the passive radio frequency identification architecture. Unfortunately, there are several factors limiting such widespread acceptance of the system, one of which is the data anomalies, such as missed, wrong and duplicate readings. These anomalies are caused due to the use of RFID passive tags to conduct the process. If this were to be corrected, high laboured tasks, such as inventory checks or supermarket shopping, can benefit greatly by cutting costs in manual labour and time.

A way to enhance the observational data to allow it to be meaningful in applications is to employ an intelligent algorithm such as a classifier that can be applied to correct ambiguous entries when given several different options to enhance the system. The *Artificial Neural Network* trained using back-propagation or an evolutionary algorithm has been shown in various applications as an effective and efficient classifier

that can be utilised to discover the correct output when given various situations.

Common methodologies that attempt to correct data anomalies found in RFID observational data either use filtration applied to streamed data at the reader level or correction algorithms applied to data that has been already stored. In literature, a filtration algorithm has been mentioned as ideal for most situations that do not need analysis of the whole data set. However, this limits its cleaning potential to only a fraction of the anomalies. Also, most deferred algorithms lack intelligence which need to be applied when ambiguous situations occur. Additionally, in previous work we utilised several intelligent methods such as Bayesian Networks and Non-Monotonic Reasoning applied at a deferred stage to correct missed data anomalies. With regards to the scope of our work, we focus directly on RFID technology and previous methodologies that have been proposed to correct such errors.

In this work, we propose the use of an Artificial Neural Network (ANN) with an analysis technique to correct stored observational data from a static RFID-enabled environment. In essence, we are attempting to correct the fundamental problem of missing spatial-temporal data by replacing it with generated values based on the shortest path between the two gaps. We have examined the most effective training algorithm of two prominent ANN algorithms and found that the Genetic Algorithm training provided a thorough cleaning result. We also compared the neural network, found from the training experimentation, with two Deferred Bayesian Networks and found that the ANN provided a higher cleaning rate.

The remainder of this paper is organised as follows: In Section 2 we provide a brief survey of Radio Frequency Identification and Artificial Neural Networks. Section 3 summarises the state-of-the-art methodologies that are currently employed to correct observational data. An analysis of our methodology is provided in Section 4, which will be broken up into the analysis phase, correction phase and Neural Network of our algorithm. The exact nature of our experimentation is explained in Section 5 followed by results and analysis in Section 6. Finally, in Section 7, we conclude our work and outline future work.

## 2 Background

Since the introduction of the passive radio frequency identification (RFID) system, the scientific community has been attempting to correct the significant anomalies present in the architecture that prevent it from being widely adopted in real-world scenarios. Prevalent anomalies present in the data warehouse include observational records that are missed within the data collection cycle. To combat this, a higher

level of intelligence is needed to be applied to the data sets to impute and restore the missed data. One such method is a classifying algorithm which would be utilised to discover the correct missing data, such as an *Artificial Neural Network* (ANN). An ANN operates by reading inputs, which it then trains upon to classify to achieve the desired output. By the end of its training, it is hoped that the ANN will be able to classify the input into a correct output.

## 2.1 Radio Frequency Identification

Radio Frequency Identification is a collection of wireless devices utilised together to recognise large sums of items. The system has already employed in several real-world applications such as postal-tracking, aviation-parts identification and baggage handling at airports (Collins 2004). As seen in Figure 1, there are four main components that combine to effectively create a RFID system: the reader, tag, middleware and data warehouse. The reader will search the area the user desires to record tag's identifier that responds, the time that this reading took place and the identifier of the reader. The tag will then transmit its unique identifier named the Electronic Product Code (EPC) to the reader. These recorded observations are then delivered to the middleware to perform immediate correcting techniques and store the information into a data warehouse (Chawathe et al. 2004).

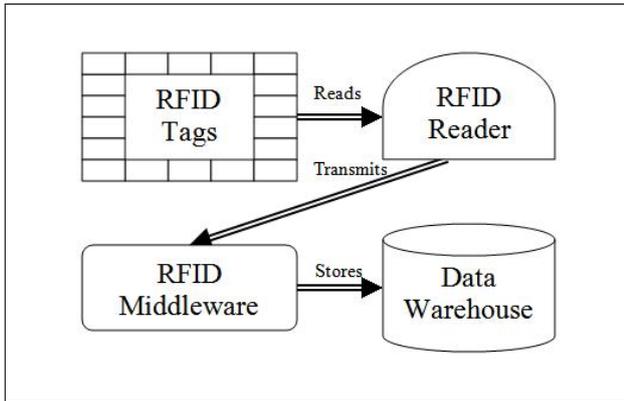


Figure 1: A high level diagram of how data is passed through the passive RFID architecture.

When employing a RFID system, there are several different tags utilised to allow items to be identified. These include passive, semi-active and active tags. The passive tag is the cheapest tag due to the lack of any battery source which also theoretically allows it to have an indefinite lifespan. It extracts its power from the electro-magnetic pulse generated from the reader and uses it to fire its unique identifier back to the reader. The semi-active tag utilises a battery as a power source but only to extend the range of the transmission. It still utilises the pulse emitted from the reader to send its identifier back which results in it having a 5-10 year lifespan. The last tag is the most advanced and expensive identifier on the market, the active tag. It utilises the battery to both extend its reading range and send its identifier. This, regrettably, results in the active tag having a lifespan of only 1-5 years long (Chawathe et al. 2004).

Unfortunately, there are severe problems that hinder the wide scale deployment of RFID systems across the commercial sector. The four main problems are identified as follows: the data generated is low level, as seen in Table 1; the data is error prone; the data generated is received in high volumes and the spatial-temporal nature of the data is complex. The data

generated from the readers are in the form of a tag identifier, timestamp and reader identifier. Unfortunately, without transforming the data to higher level information, such as events, using additional tag and reader information, the information is useless (Khoussainova et al. 2007). Most RFID systems, especially the architectures that rely on passive tags, generate three core anomalies: wrong, missing and duplicate data. Due to the continuing scan of tags every second or less, the data storage suffers from the problem of massive intakes of data. It has been stated that Wal\*Marts in America which have employed the use of RFID technology generate 7TB of data daily (Raskino et al. 2005). With the increasing developments of passive technology, there is also the issue of complexity with regards to spatial and temporal data (Wang & Liu 2005). For example, if there is small hand-held reader utilised to scan different locations, the spatial location of the reader will also constantly be different.

Tag EPC	Reader ID	Timestamp
t1	r1	2009-08-15 14:05:08.002
t2	r1	2009-08-15 14:05:08.002
t1	r2	2009-08-15 14:45:54.028
t2	r1	2009-08-15 14:05:08.002
t1	r3	2009-08-15 15:02:06.029
t2	r1	2009-08-15 15:02:06.029
t1	r4	2009-08-15 15:18:49.016
t2	r1	2009-08-15 15:18:49.016

Table 1: A table populated with sample RFID Data containing the information of EPC, Reader and Timestamp. Please note that actual RFID data would contain unique identifiers for the tag EPC and reader identifier.

As mentioned above, the data anomalies within the generated data include missed, wrong and duplicate observational (Jeffery et al. 2006)(Bai et al. 2006). Without the correction of these anomalies, the passive RFID architecture will never be utilised to its full potential as a cost effective and efficient means of wireless identification of large sums of items. Of the three anomalies, missing observations may be considered the hardest to restore as the data is not recorded into the database and, therefore, does not have as much analytical information when attempting to correct the records. It has been estimated that reader only captures 60%-70% of the data that was supposed to be recorded (Floerkemeier & Lampe 2004).

## 2.2 Artificial Neural Networks

Artificial Neural Networks (ANN) is a classifying tool that is modelled after the biological neural network found in the brains of animals. There are four main components of a biological neuron in the brain as seen in Figure 2: the dendrites that receive the information; the cell body that reacts to the information given a certain threshold; the axon that is the output of the neuron and the synapses which connects various dendrites to axons. The brain itself possesses a large sum of neurons which all work in unison to form a neural network, it is estimated that in humans, the amount of neurons presently in the brain range from 10 billion to 1 trillion (Williams & Herrup 1988).

The ANN operates by receiving inputs, manipulating them through the use of weights between hidden units and hidden layers in each neuron to provide an output, as shown in Figure 3 (McCulloch & Pitts 1943). Its main advantage is that it is able to generalise information and provide a relatively correct answer. There are many ways through which to train the weights of an ANN through means of

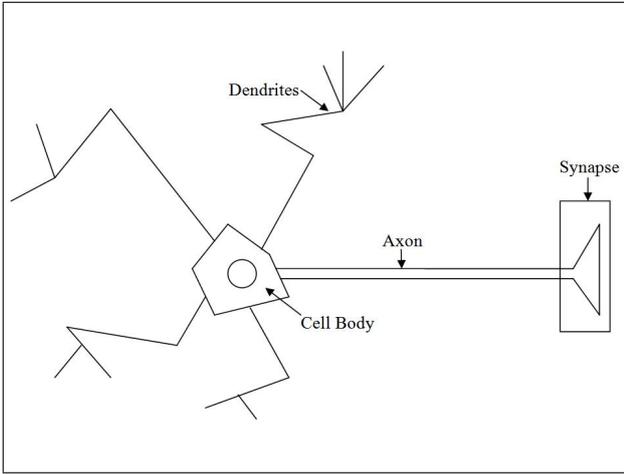


Figure 2: A diagram representing a biological neuron's structure found in living organisms.

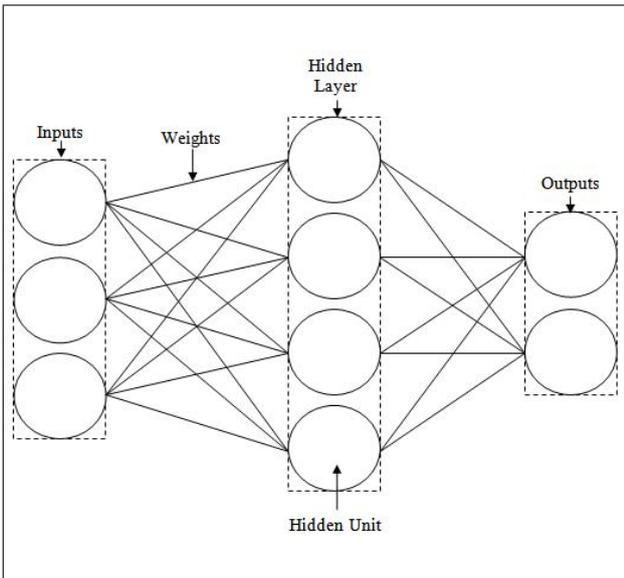


Figure 3: A high level diagram of the various components that create a feed-forward perceptron neural network.

algorithms such as Back-Propagation or Evolutionary Neural Networks. An artificial neuron pictured in Figure 4 is composed of various inputs that are manipulated by weights which are summed to find the neural activity for that specific neuron. Activation Functions are then utilised to change the output to a desirable answer that can be used in the applications. There are two current prominent activation functions: the hard limiter, which will either classify the output as positive one or below zero, and the sigmoidal function, which applies the equation found in Equation 1 to the output.

$$f(x) = 1/(1 + \exp(-x)) \quad (1)$$

### 2.2.1 Back-Propagation

The Back-Propagation algorithm utilises the overall error found in the actual outputs when compared with the desired outputs to manipulate the weights of the neural network. As seen in Figure 5, the back-propagation algorithm procedures are as follows (Rumelhart et al. 1986); the user presents the training sets to the network in the form of inputs and desired outputs; the network calculates the error rate

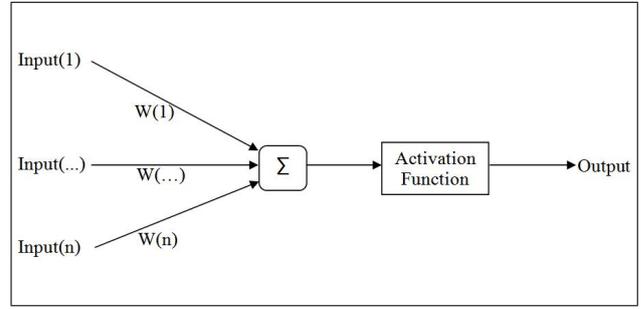


Figure 4: A diagram representing an artificial neuron's structure found in artificial neural networks.

between the actual output and desired output and manipulates accordingly to obtain a lower error rate on the next round. The algorithm is stopped with only two criteria. The first is a user specified number of iterations and the second is a Root-Mean-Square (RMS) error threshold also specified by the user (Blumenstein et al. 2007).

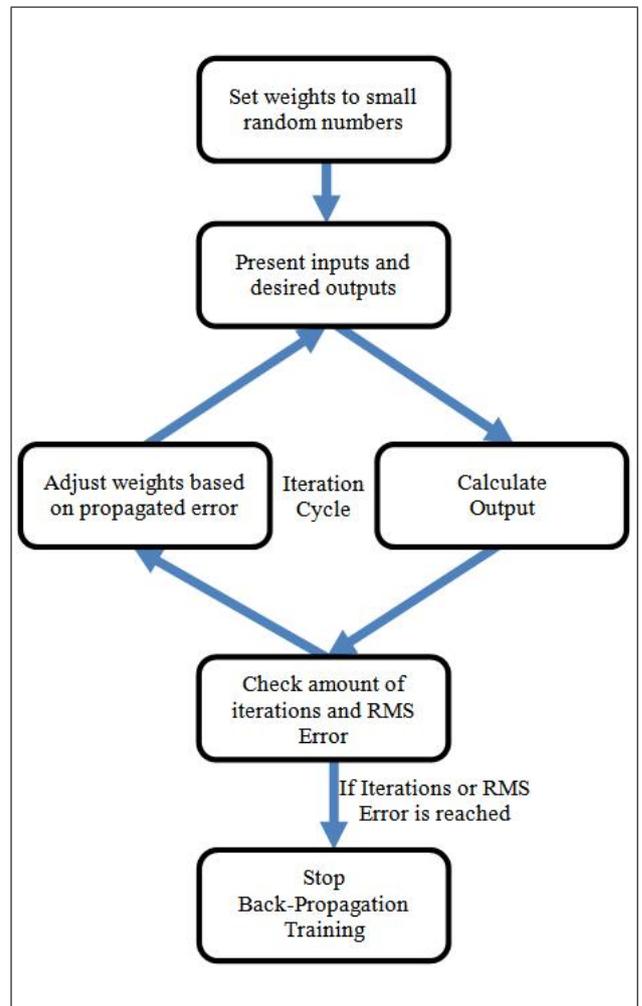


Figure 5: A high level flow diagram of the processes necessary to train a neural network using the back-propagation algorithm.

### 2.2.2 Evolutionary Neural Networks

An Evolutionary Neural Network is an ANN that utilises a genetic algorithm modelled after the theory of evolution to manipulate the weights accordingly (Holland 1975). It does this by initialising chromosomes that have information of all the weights of the

neural network to small values and finding the most correct values (Rooij et al. 1996). A certain threshold will be used to delete the remaining chromosomes and proceed to cross over the fit chromosomes to increase the population to its original size. Additionally, a mutation will occur on a small selected amount of the population to avoid local minima. The stopping criteria is a user-specified correctness threshold or number of generations for the chromosomes. The necessary processes and order in which they occur may be viewed in Figure 6 (Cha et al. 2008).

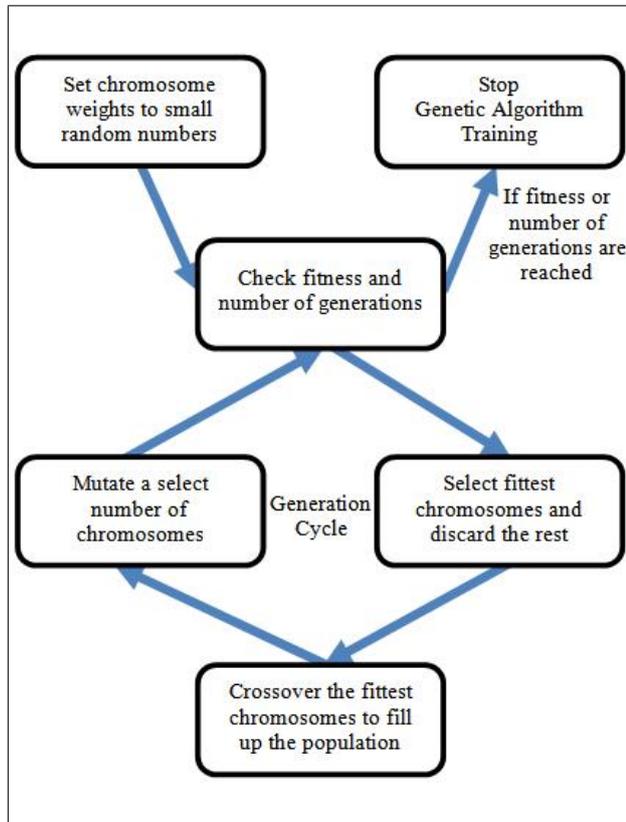


Figure 6: A high level flow diagram of the processes necessary to train a neural network using a genetic algorithm.

### 3 Related Work

The data anomaly problem of passive RFID records have been studied extensively throughout recent years with the main goal to correct the observations to allow it to be used in higher business processes. There are two main ideologies within literature. The first is to filter the incoming records when recording the data and the second is to apply correction algorithms which enhance the integrity of the observations in the stored data warehouse. Most filtration processes centre around utilising anti-collision algorithms (Shih et al. 2006) as the tags are read at the reader. Other methodologies employ rules that have been specified by the user to correct the data when it is in the database (Rao et al. 2006).

From examining the respected literature regarding these related works, we have found that the filtering and rule based methodologies sacrifice analytical knowledge and intelligence respectively which we believe is crucial to thoroughly enhance the data to its maximum potential of integrity. The filtration methods are applied at the edge and, therefore, lack analytical knowledge gained from the previous data entries and future data entries that may be examined within

the data warehouse after the recording phase. Additionally, the rule-based approach is specified by users which may allow for logical errors within the protocols and lack the intelligence needed to cope with ambiguous observations. In previous work, we have utilised Non-Monotonic Reasoning (Darcy et al. 2007), Static Bayesian Networks (Darcy et al. 2009b) and Evolutionary Bayesian Networks (Darcy et al. 2009a) to combat these drawbacks.

## 4 Methodology

The core functionality of our system which is shown in Figure 7 has been divided into two phases, the analysis phase and the correction phase. As the name implies, the analysis phase consists of an algorithm that sweeps through the RFID readings finding any missed readings that should be present. If said anomalies are found, the algorithm records various attributes regarding the gap of knowledge which are then passed on to the correction phase. The correction phase is where the system attempts to seek out the ideally correct combination of readings to impute the gapped knowledge of data. It is the goal of the system at this time to substitute the most correct data that the system can generate to allow for higher business processes to be conducted.

### 4.1 Analysis Phase

---

**Algorithm 1:** The algorithm for the Analysis Phase of the methodology. Its purpose is to discover the various algebraic values needed and find the inputs needed for the neural network.

---

```

foreach Tag do
  Start Counter at first Observation's
  Timestamp.
  foreach Reading do
    if Counter  $\neq$  Current Reading
    Timestamp then
      Record the current Reader (b) at the
      start of the Gap and the Reader
      before current (a).
      Go to the next recording after the
      Gap.
      Count the amount of Missed
      Readings in the Gap (n).
      Record the current Reader (c) at end
      of Gap and the Reader after the
      current (d).
      Find the Shortest Path between b
      and c then count how many
      observations are present (s).
      Record (a), (b), (n), (c), (d), (s) and
      the Shortest Path.
      Determine if (a==b), (b==c),
      (b $\leftrightarrow$ c), (c==d), (n==(s-2)),
      (n<(s-2)) and (n>(s-2)). Record
      these values as the inputs for the
      Neural Network phase.
    end
  end
end
  Pass the values to the Correction Phase and the
  analysis to the Neural Network phase.

```

---

The analysis phase was originally conceived from previous work in which observational data is analysed attempting to uncover several key factors that reflect the nature of the missed readings. The pseudo code version of the algorithm used to perform this phase

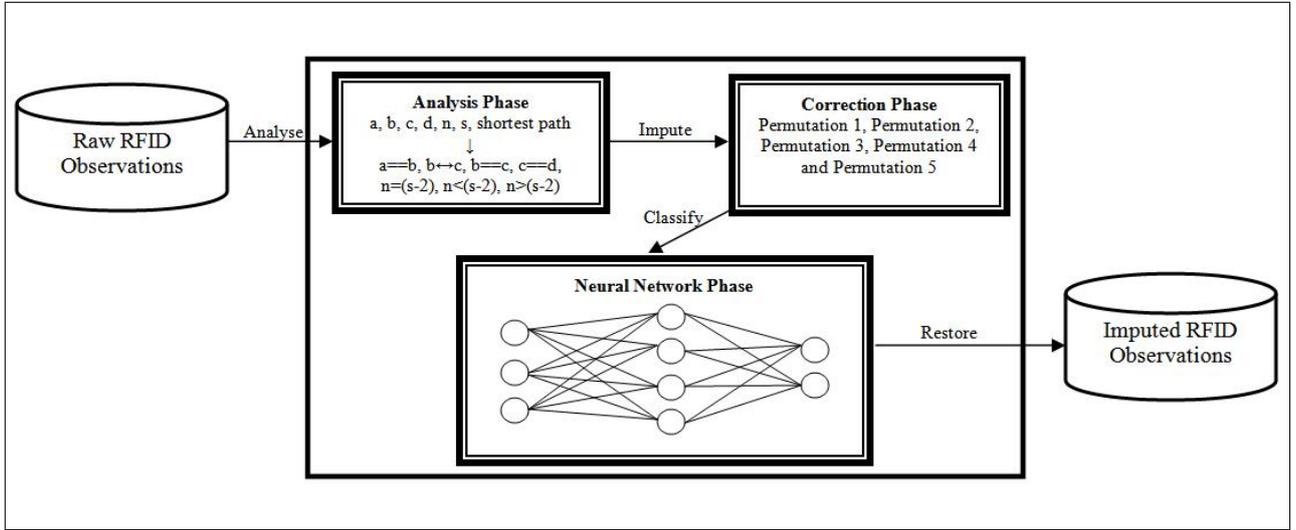


Figure 7: A high level diagram of the architecture for the methodology we have proposed which utilises intelligent analysis and a neural network to correct missed observational data.

may be found in Algorithm 1. The phase involves first separating the amalgamation of readings into individual tag streams by dividing the observations according to the identifier. After this division has been completed, we attempt to find gaps in the data that are found due to a missing periodic timestamp. We have devised our scenario with the assumption so that the readers periodically scan for readers within the vicinity of it. Additionally, it is crucial that our methodology is granted access to the map data of the readers in the static environment to impute the most likely readings.

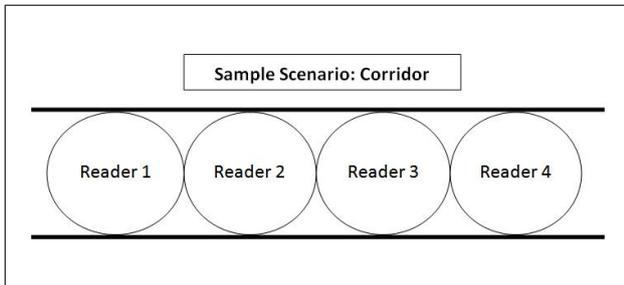


Figure 8: A sample scenario which will provide an example of how the methodology operates. The corridor environment is static with walls preventing people from not being detected from readers one to four in this order.

Once a gap of knowledge is found in the records, we analyse the surrounding readings to find six core values: the reader two observations and one observations before the gap (a and b); the amount of the missed reading (n); the reader one and two observations after the gap of knowledge (c and d respectively); and the minimum amount of readings that are required to join readings b and c utilising the map data (s). Additionally, the shortest path will be required to determine the minimum amount of readers needed to bridge the gap. For example, if there is a basic map of a corridor in which readers 1-4 are connected in a linear line as shown in Figure 8, then the data analysis algorithm would detect the values as shown in Figure 9.

The analytical phase requires that the information it had just discovered be then translated into meaningful analytical information. To accomplish this, the process will find the following mathematical operations which we believe can be utilised to determine

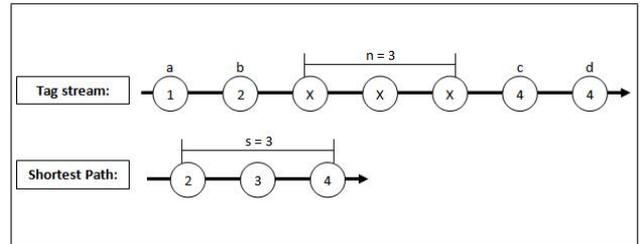


Figure 9: The variables and shortest path which are needed for the imputation process from the sample scenario which have been placed onto the sample scenario's tag stream.

the most correct readings to fill the gaps of knowledge. These analytical operations include: finding if readers "a" and "b" are equal ( $a==b$ ); determining if readers "b" and "c" are relatively close to each other one according to the map data ( $b<->c$ ); discovering if the readers of "b" and "c" are equal ( $b==c$ ); finding if readers of "c" and "d" are equal ( $c==d$ ) and discovering if "n" is equal to, less than or greater than "s" minus two ( $n==(s-2)$ ), ( $n<(s-2)$ ), ( $n>(s-2)$ ). The reasoning behind subtracting two from value "s" is that the shortest path will include the values of "b" and "c" which are not necessarily a part of the missing gaps of knowledge. All of these analytical boolean variables are then passed on to the correction phase which utilise it to seek out the most ideal imputed reader values.

## 4.2 Correction Phase

The Correction Phase, whose pseudo code may be viewed in Algorithm 2, is where the possible permutations of the imputed data is created. It relies on the information that was obtained from the analysis processes to use as evidence to determine the most correct data. When the information is passed to this phase, five possible combinations of the data which we have termed "Permutations" will be generated. These may viewed in Figure 10. The first permutation will generate the missed readings to all have the value of reader "b". The second permutation is a mirror to the first in that it will substitute all the missed values with the reader "c". The third requires the use of the shortest path generated before to insert it into the middle of the missed observations. Any ad-

---

**Algorithm 2:** The algorithm for the Correction Phase of the methodology. Its purpose is to develop the permutations needed as the output for the neural network.

---

Receive the algebraic values of the missing data from the Analysis Phase.

**foreach** *MissingRecord* **do**

Create an Array of size “n”.

Generate the first permutation by using the value of “b” substituted for each recording.

Generate the second permutation by using the value of “c” substituted for each recording.

Generate the third permutation by inserting the shortest path in the middle of the array and substitute the values of “b” and “c” on the left and right side of the array to additional missing gaps.

Generate the fourth permutation by inserting the shortest path on the right of the array and substitute the value of “b” on for all remaining gaps.

Generate the fifth permutation by inserting the shortest path on the left of the array and substitute the value of “c” on for all remaining gaps.

**end**

Pass the permutations to the Neural Network phase.

---

ditional missed values will have reader values “b” or “c” substituted where it is in close proximity.

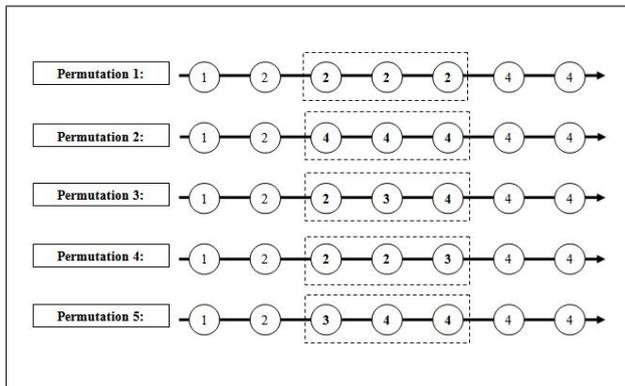


Figure 10: The permutations generated for the scenario which are to be inserted back into the data storage as imputed readings.

The fourth permutation will utilise the shortest path again, but place it on the latter missed readings rather than the middle. All missing values that are on the former side of the missing data will be substituted with the reader value “b”. The final permutation generated will once again be a mirrored duplicated of the fourth in that the shortest path will be inserted to the former part of the gap of knowledge and the reader value “c” will be substituted for any additional missed readings. We believe that these permutations represent the most likely candidates of the data in that each one finds the shortest route between the starting and destination points which we also believe is likely for the average tag. After these possible permutation combinations have been derived each one, with the addition of the extra analytical data, is then forwarded onto the ANN to determine what it believes is the most likely candidate for restoration.

### 4.3 Artificial Neural Network Phase

We have employed the use of an Artificial Neural Network (ANN) in the last phase as a means to classify which permutation is the most correct given the analysis data ascertained from the previous two phases. The ANN accepts seven binary inputs to reflect the analysis data (a==b, c==d, etc.) and has five binary outputs to reflect that the permutations that have been found. The ANN itself will be comprised of nine hidden units and one hidden layer as seen in Figure 11.

The reason why we have chosen this configuration is that we wish to have a number of units greater than the number of inputs and we believe that increasing the number of hidden layers will not necessarily increase the accuracy. We have also applied a momentum term and learning rate whose values are be 0.4 and 0.6 respectively to avoid local minima and network paralysis. Each input and output value will not be 1 and 0 as this may not yield a very high classification rate, instead we will use the values of 0.9 and 0.1 respectively. We will set the stopping criteria as both the RMS error threshold when it reaches below 0.1 and 1000 iterations. We have also utilised the sigmoidal activation function to derive our outputs. The pseudo code for both the back-propagation and genetic algorithm training methods can be found Algorithms 3 and 4 respectively.

---

**Algorithm 3:** The algorithm for training the Neural Network Phase utilising the Back-Propagation Algorithm. The goal of this algorithm is to modify the weights using forward and backward passes until the network has been trained to determine the correct outputs.

---

Initialise the weights to small random values.

Present the training set to the neural network.

**foreach** *Iteration* **do**

Determine the actual output of all the neurons (Forward Pass).

Check stopping criteria (RMS Error and Iterations).

**if** *Stopping Criteria is True* **then**

Break the algorithm and store the weights.

**end**

Determine the error terms.

Adjust the weights using the learning rate, momentum and error terms (Backward Pass).

**end**

Store highest performing network configuration.

---

## 5 Experimental Evaluation

To determine the maximum accuracy of our methodology and compare it to other state-of-the-art applications, we have devised two experiments. We have run both of these experiments on a standard computer environment at our institute to demonstrate that the system may be run in a typical environment. The two experiments have been designed to determine the best training algorithm to configure the neural network, and compare the said network to a deferred bayesian network implementation to find the significance of methodology.

### 5.1 Environment

The computational environment of our experiment consists of utilising the C++ language to write the

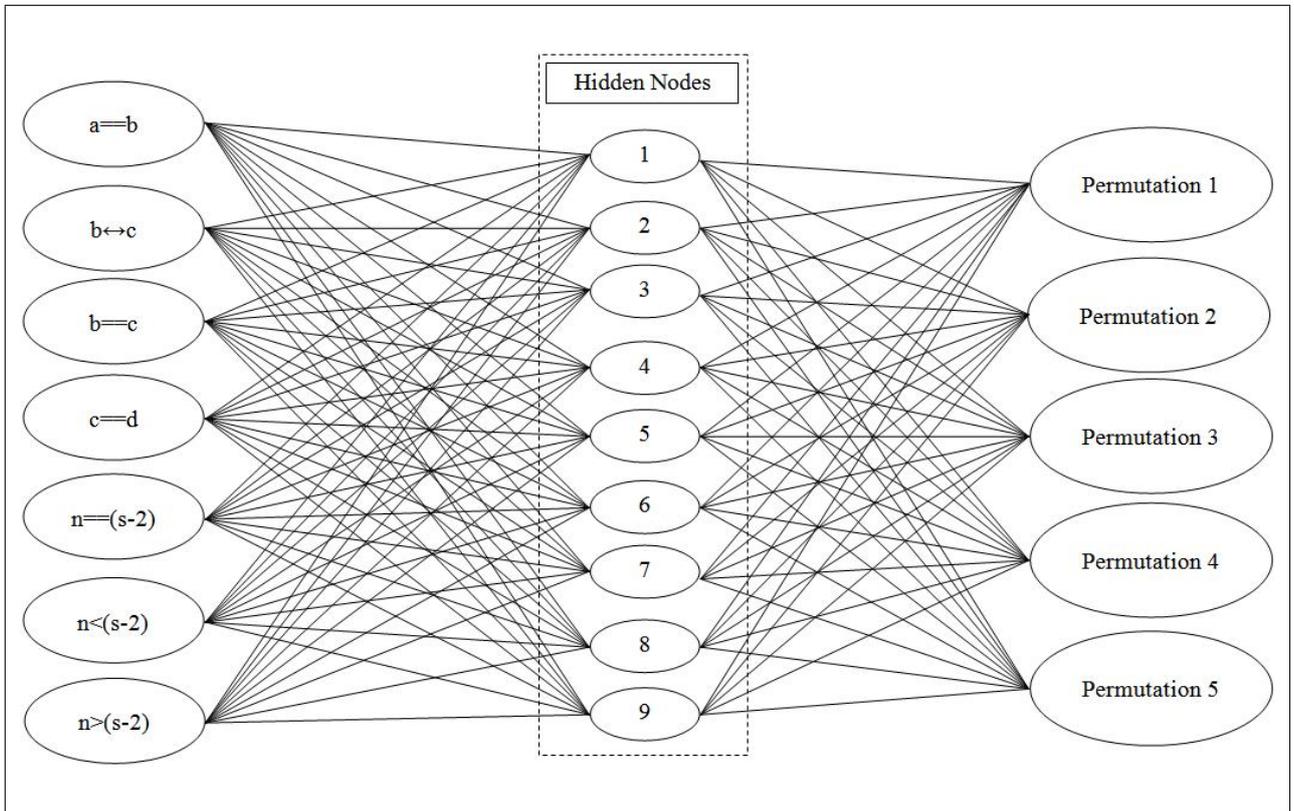


Figure 11: A high level representation describing the structure of the neural network we are utilising within our work. The 7 inputs include all the analytical information gathered from the analysis phase, whereas the 5 outputs are all five permutations which were generated during the correction phase. We have also employed 9 hidden nodes inside the network which results in the sum of 108 weights that connect each of the nodes to either the inputs or outputs.

---

**Algorithm 4:** The algorithm for training the Neural Network Phase utilising the Genetic Algorithm. The goal of this algorithm is to set the weights to a point they are able to chose the correct permutation when given various training sets.

---

```

Initialise the weights to small random values.
Present the training set to the neural network.
foreach Generation do
  Evaluate and order the chromosome
  according to fitness (the classification rate).
  Check stopping criteria.
  Check stopping criteria (Generations).
  if Stopping Criteria is True then
    Break the algorithm and store the
    weights of the highest performing
    chromosome.
  end
  Delete half the least fit population members
  and breed two random fitter chromosomes
  to fill the amount of population.
  if Child chromosome is already present in
  population then
    Discard child chromosome.
    Breed two different random parents.
  end
  Apply mutation to select members of the
  chromosome by resetting the weights to
  random values - 1% the top 10% fittest
  population members, 5% for other
  chromosomes.
end
Store highest performing network configuration.

```

---

code and execute it using Microsoft Visual C++ 6.0. It has been run on a computer that has the Windows XP operating system that runs Service Pack 3 Intel (R) Pentium (R) 4 CPU 2.79 GHz with 2.00 GB of RAM.

## 5.2 Experiments

We have utilised two different types of experiments to determine the significance of our methodology. The first experiment which we labelled the “Configuration Experiment”, we have set up is designed to determine which of the training techniques yields a higher accuracy for the neural network. The two training algorithms we will compare are a back-propagation and a genetic training algorithm as we have found that they are prominent in the field of training neural networks. The second experiment which we labelled the “Significance Experiment”, we have decided to conduct involved proving the significance of the most accurate neural networks result from the configuration experiment when compared with the accuracy of a deferred bayesian network approach. We chose a deferred bayesian network as it has been shown in the past to obtain a high accuracy rate and is directly comparable to our deferred neural network approach. The training and testing sets will consist of permutation scenarios in which the worst case occurs. In reality, this scenario has an extremely low chance of occurring as our algorithm has been designed so that where the missed readings are purely random and short in size, which frequently happens in physical applications, every permutation generated will be correct.

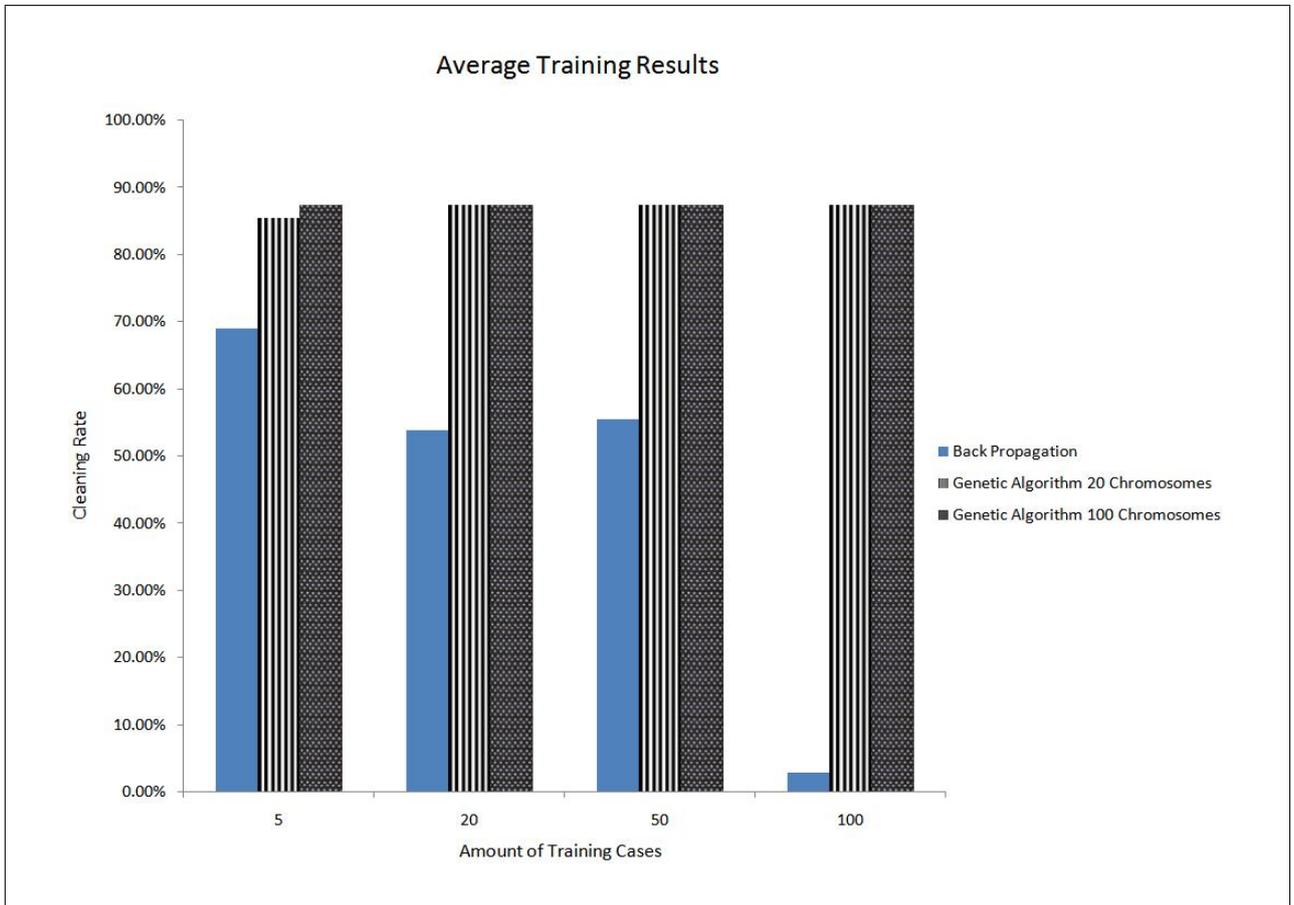


Figure 12: A bar graph depicting the averaged results of the three training algorithms versus the cleaning rate to determine the most effective network configuration for our methodology. The solid bar represents the Back-Propagation algorithm, the striped bar represents the Genetic Algorithm with 20 chromosomes applied and the dotted bar represents the Genetic Algorithm with 100 chromosomes applied.

## 6 Results and Analysis

In this work, we have proposed a methodology that harnesses a powerful data analysis technique coupled with a neural network to correct missing observations in stored RFID data. To this end, we have devised two experiments to determine the best network configuration to use when comparing prominent training algorithms and analysed the effectiveness of both it and two Bayesian Networks. We have recorded and evaluated the results of these experiments directing focus to the rate of cleaning in choosing the correct permutation to restore the observational data.

### 6.1 Configuration Experiment

The Configuration Experiment has the goal of seeking out the neural network training algorithm that yields the highest clean rate. The two training methods used for comparison are the back-propagation and genetic algorithms. The training set utilised in this experiment is comprised of every possible combination of the inputs and their respective outputs which amount to a total of 128 entries. We have conducted tests upon three different training algorithm setups, the first is the back-propagation algorithm and the other two are genetic algorithms that use 20 and 100 chromosomes to find the optimised solution. Additionally, we conducted each experiment in three trials to further generalise our results. The algorithm that had the hardest time finding the correct configuration was the back-propagation algorithm when it iterated for 50 and 100 times, earning it 1.56% classification rate. The trainer that performed the best was the

Genetic Algorithm both using 20 and 100 chromosomes in every test and iteration number excluding the 20 chromosome configuration which lasted for 5 generations.

The analysis we performed on these results consisted of us graphing the average of our findings into a bar graph to illustrate the difference in algorithms. As shown in Figure 12, on average the Neural Network Genetic Algorithm performed the best obtaining an 87.5% cleaning rate. Unfortunately, as discovered before in the results, the back-propagation algorithm performed the weakest within the three algorithms. This is especially present when it is attempting to clean after being trained for 100 iterations. We believe the poor results of the back-propagation algorithm was due directly to over-training the network. We noticed that there was also a particularly low result when attempting to train this algorithm for 50 attempts in trial 2. However, it wasn't until the 100 iteration training that we could clearly see the effects of the training routine on the average cleaning rate.

As a general observation from these averaged results, we can clearly see that, with the accepting of the 5 iterations experiment, each training result had both genetic algorithms obtaining the same cleaning rate. After examining the weights of both the 20 and 100 chromosome genetic algorithm configurations, we found that although the same result was obtained, the networks that had been generated were different. We believe that this due to there being many certain networks that may be utilised to find an average maximum of classification from the data set which happens to be 87.5% (the highest cleaning rate achieved). Furthermore, we believe that the results obtained from

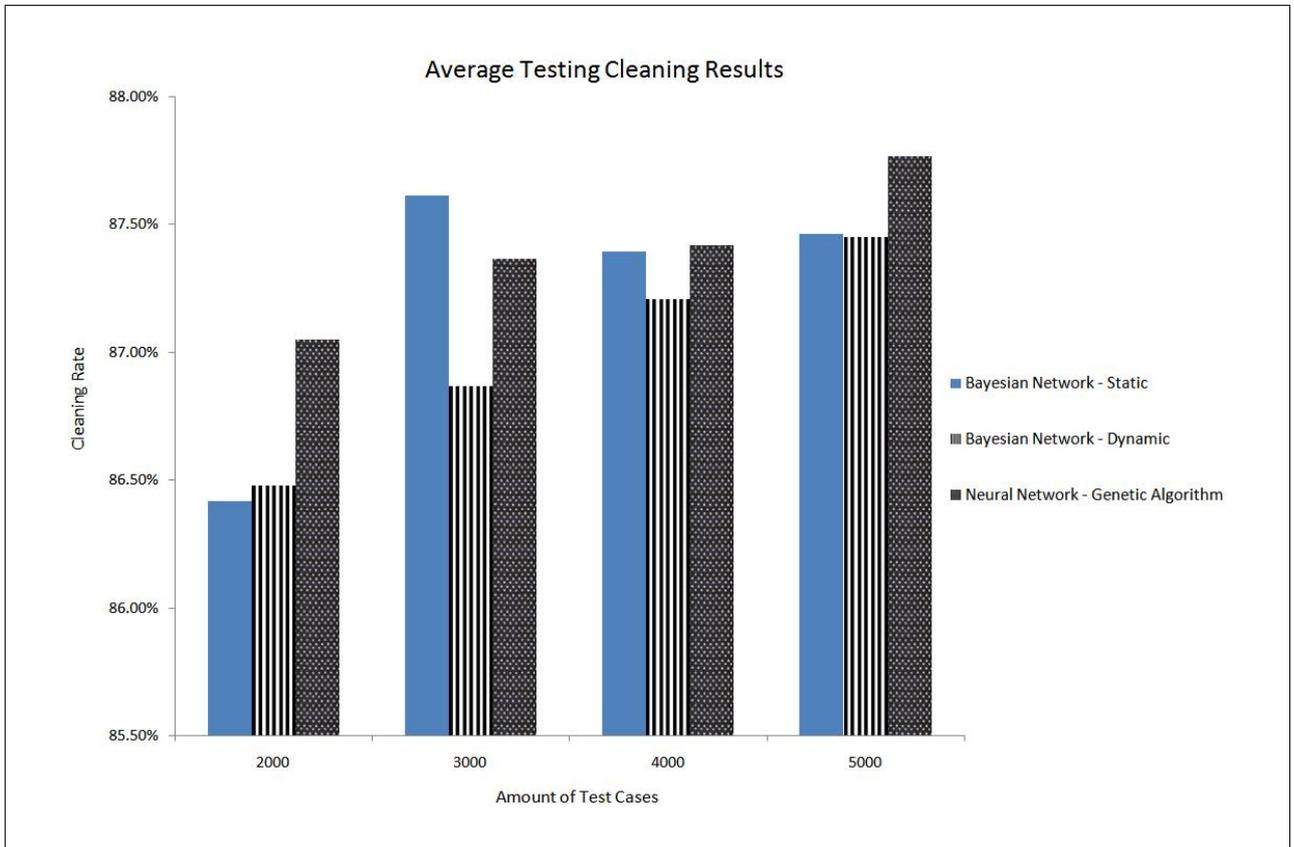


Figure 13: A bar graph depicting the averaged results of the three testing algorithms being utilised to depict the significance of the Neural Network methodology. The solid, striped and dotted bar represents the static bayesian network, dynamic bayesian network and neural network trained by a genetic algorithm respectively.

the first experiment were lower for the 20 chromosome network due to there not being enough generations to find this maximum classification network. The reason we believe why the 100 chromosome was able to find this maximum classifying network in such a short amount of time is that it had the advantage of having five times as much weight configurations to evaluate within its population.

## 6.2 Significance Experiment

The Significance Experiment was created to show that our algorithm can compete with other deferred RFID cleaning algorithms. We utilise the network that achieved the highest record in the Configuration Experiment to compare it to the cleaning rate of two deferred bayesian networks, one dynamic and the other static. Each algorithm has been tested upon four different data sets that contain varying amounts of missing data. These data sets either have 20%, 30%, 40% or 50% missed readings from a 10,000 record data set. The inspiration for the varying amounts of missing data is to properly emulate the ideal that at any given time; passive readers will only record 60%-70% of the total observations. The testing set has been based on the worst possible scenario where only one permutation is correct for each set of data whereas, usually, this occurs minimally.

The configuration that was utilised for this Neural Network was a Genetic Algorithm that was used to train 100 chromosomes for 100 generations. We chose the genetic algorithm due to its high cleaning rate, and chose one that had a relatively more evolved chromosome that the other algorithms. The highest performing algorithm in this experiment was the Neural Network which obtained an 88.36% cleaning rate when correcting 5000 test cases in its first trial.

The algorithm obtaining the least cleaning rate was the static bayesian network which only obtained an 85.95% rate of correctness. As with our previous experiment, we performed three trials upon the algorithms to illustrate how they react to the data.

As with the Configuration Experiment, we have also generated a bar graph to illustrate how effective the average cleaning rates of the three algorithms is. As seen in Figure 13, the highest average cleaning rate was obtained by the Neural Network which was trained by the Genetic Algorithm. Also of note with regard to the highest cleaning is that it occurred when there is 50% missing data which is what we had planned to address due to there being a more likely chance of consecutive missed readings. The lowest achieving cleaning algorithm was both the Static and Dynamic Bayesian Networks. These low readings occurred when there was 20% of the data missing.

Also reflected from the results is that the static bayesian network performed at their peaks during the 30% missing data test cases whereas the dynamic bayesian network performed best cleaning the 50% missing records. Both bayesian networks were beaten in cleaning rate in every test case except for the 30% data set. We believe that this is due to the neural network not being able to clean the data sets with little missing records as effectively as the sets with large amounts of missing data. This is also present within the data set that only contains 20% missing data as the neural network achieved its lowest average performance.

## 7 Conclusion

In this work, we have researched the problem of missed observations within the data warehouse of a

Radio Frequency Identification system utilising intelligent data analysis and an Artificial Neural Network. We have created a methodology that analyses the RFID data to search for missing record anomalies which will have several permutations of possible imputed value sets to be inserted into a data storage. Then, from the utilisation of a neural network that has been trained with a back-propagation algorithm and two genetic algorithms, we determine the most correct permutation to be inserted back into the data storage. We then compared the highest performing network configuration with that of two bayesian networks which have both been trained to correct missing RFID data and found that our neural network obtains a higher cleaning rate.

More specifically, this study makes the following contribution to the field:

- We proposed a deferred methodology that utilises both a Neural Network and intelligent data analysis.
- We applied this methodology to restore missed RFID readings which have not been recorded in the database.
- We studied the accuracy of the resulting networks from two leading ANN training algorithms, back-propagation and genetic algorithms, to determine which would be best suited as an RFID missed readings classifier. This concluded with us finding that the genetic algorithm training provided a superiorly accurate imputed data set.
- We compared our new methodology to two bayesian network methodologies which has also been used to impute missed observations. We found that our ANN algorithm yields a higher cleaning rate especially where the data set is missing approximately half its records.

With regards to future work, we would like to address the application of other training algorithms to our ANN to see if the resulting network will yield a more accurate imputed readings. We would also like to record the effectiveness of the network when the momentum, stopping criteria, hidden units and hidden layers have been manipulated. Additionally, we would also like to attempt to modify the feature extraction process of analysed values used within the Neural Network and possibly apply it to situations outside of RFID applications. Finally, we would like to develop an intelligent data analysis algorithm coupled with the classifiers we have researched to correct other anomalies present within an RFID data set.

## Acknowledgment

This research is partly sponsored by ARC (Australian Research Council) grant no DP0557303.

## References

Bai, Y., Wang, F. & Liu, P. (2006), Efficiently Filtering RFID Data Streams, in 'CleanDB'.

Blumenstein, M., Liu, X. Y. & Verma, B. (2007), 'An Investigation of the Modified Direction Feature for Cursive Character Recognition', *Pattern Recognition* **40**(2), 376–388.

Cha, D., Blumenstein, M., Zhang, H. & Jeng, D.-S. (2008), A Neural-Genetic Technique for Coastal Engineering: Determining Wave-induced Seabed Liquefaction Depth, in 'Engineering Evolutionary Intelligent Systems', pp. 337–351.

Chawathe, S. S., Krishnamurthy, V., Ramachandran, S. & Sarma, S. E. (2004), Managing RFID Data, in 'VLDB', pp. 1189–1195.

Collins, J. (2004), 'Boeing Outlines Tagging Timetable [online]', RFID Journal. Available from: <<http://www.rfidjournal.com/article/view/985/1/1>> [Accessed: 5th November 2008].

Darcy, P., Stantic, B. & Derakhshan, R. (2007), 'Correcting Stored RFID Data with Non-Monotonic Reasoning', *Principles and Applications in Information Systems and Technology (PAIST)* **1**(1), 65–77.

Darcy, P., Stantic, B. & Sattar, A. (2009a), Augmenting a Deferred Bayesian Network with a Genetic Algorithm to Correct Missed RFID Readings, in 'Malaysian Joint Conference on Artificial Intelligence (MJCAI 2009)', pp. 106–115.

Darcy, P., Stantic, B. & Sattar, A. (2009b), Improving the Quality of RFID Data by Utilising a Bayesian Network Cleaning Method, in 'Proceedings of the IASTED International Conference Artificial Intelligence and Applications (AIA 2009)', pp. 94–99.

Floerkemeier, C. & Lampe, M. (2004), Issues with RFID usage in ubiquitous computing applications, in A. Ferscha & F. Mattern, eds, 'Pervasive Computing: Second International Conference, PERVASIVE 2004', number 3001, Springer-Verlag, Linz/Vienna, Austria, pp. 188–193.

Holland, J. (1975), *Adaption in Natural and Artificial Systems*, University of Michigan Press, Cambridge, MA, USA.

Jeffery, S. R., Garofalakis, M. N. & Franklin, M. J. (2006), Adaptive Cleaning for RFID Data Streams, in 'VLDB', pp. 163–174.

Khoussainova, N., Balazinska, M. & Suciu, D. (2007), 'Probabilistic RFID Data Management', *UW CSE Technical Report UW-CSE-07-03-01*.

Mcculloch, W. S. & Pitts, W. (1943), 'A Logical Calculus of the Ideas Immanent in Nervous Activity', *Bulletin of Mathematical Biophysic* **5**, 115–133.

Rao, J., Doraiswamy, S., Thakkar, H. & Colby, L. S. (2006), A Deferred Cleansing Method for RFID Data Analytics, in 'VLDB', pp. 175–186.

Raskino, M., Fenn, J. & Lenden, A. (2005), Extracting Value From the Massively Connected World of 2015, Technical Report G00125949, Gartner Research.

Rooij, A. J. F. V., Johnson, R. P. & Jain, L. C. (1996), *Neural Network Training Using Genetic Algorithms*, World Scientific Publishing Co., Inc., River Edge, NJ, USA.

Rumelhart, D., Hinton, G. & Williams, R. (1986), 'Learning Representations by Back-Propagating Errors', *Nature (London)* **323**, 533–536.

Shih, D.-H., Sun, P.-L., Yen, D. C. & Huang, S.-M. (2006), 'Taxonomy and Survey of RFID Anti-Collision Protocols', *Computer Communications* **29**(11), 2150–2166.

Wang, F. & Liu, P. (2005), Temporal Management of RFID Data, in 'VLDB', pp. 1128–1139.

Williams, R. W. & Herrup, K. (1988), 'The Control of Neuron Number', *Annual Review of Neuroscience* **11**(1), 423–453.