SMART JARMS – COMPUTATIONAL INTELLIGENCE IN SIMULATION

Steve Manzi, Shelia Burgess and Debbie Berry
Martin Marietta Information Systems Company
Flight Systems Group
Orlando, Florida

INTRODUCTION

The ability to simulate Electronic Combat (EC) is a vital part of networked virtual reality simulation. At Kirtland Air Force Base in Albuquerque, New Mexico, the 542nd Combat Crew Training Wing utilizes this capability to support Special Operations Forces mission rehearsal and training. Importance to National Command Authorities of successful missions rehearsed at this facility cannot be overstated. Therefore, the highest level of fidelity to the real world must be achieved during simulation.

Pursuing an upgrade to the EC simulation as proposed herein would enhance overall mission rehearsal capability. EC simulation consists of software developed in the late 1970s rehosted on modern hardware. Although adequate, improvements are possible. This study examines the EC simulation to determine where computational intelligence techniques can be applied to provide an improved solution.

The purpose of this study is to investigate two *Computational Intelligence (CI) candidates for* replacement of the Offensive Tactics Simulation. The two techniques employed are Fuzzy Systems (FSs) and Neural Networks (NNs). FSs are knowledge based systems that allow subjective manipulation of inexact concepts. NNs are an idealization of the interconnections and functions of a nervous system which mimic the brain's learning and thought processes. Initial modeling of both techniques is performed and the results reported. Implementation of either system could potentially yield performance improvements.

Overviews of real world weapon systems and the current EC simulation are provided. Then, the approach used to develop CI solutions is defined. The FS and NN solutions are examined along with implementation considerations, empirical results, and conclusions.

Weapon System Overview

Modern weapon systems are usually composed of many subsystems, three of which could be a sensor to detect targets, a guidance system to direct the weapon, and the weapon itself.

A typical example would be a radar guided surface–to–air missile (SAM) system. Radar subsystems typically exhibit a number of distinct modes which are differentiated by observable characteristics, some of which are Radio Frequency (RF), Pulse Width (PW), Pulse Repetition Frequency (PRF), and Scan Pattern. The overall weapon system state would progress from mode to mode based on specific parametrics of the ownship/weapon engagement. Some of these parametrics are:

- Range
- Altitude
- Elevation
- Occult Status
- Jamming Effectivity
- Weapon in Flight

For example, a radar guided SAM could typically progress through the modes shown (see Figure 1) during an inbound radial ownship/weapon system engagement.

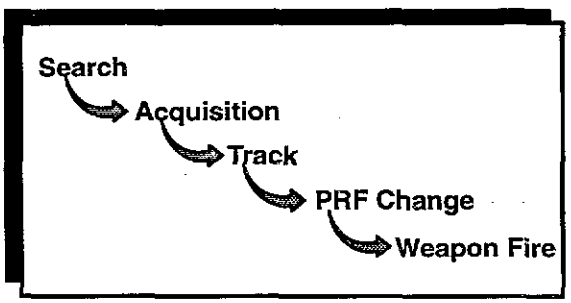

Figure 1. Mode Progression

Individual weapon systems may be joined into Command, Control and Communications (C^3) networks which affect their response. Under the conditions of open conflict the response time of individual weapon systems is typically decreased. This occurs when the search phase of the engagement is made more efficient when data from early warning systems are made available to fire control systems. Under different circumstances, components of the chain of command may exercise control over a specific encounter for political or military reasons, thus response time could increase.

Current EC Simulation

The current simulation of weapons systems is via a pseudo object oriented approach in which software entities known as JARMS (Jammer, Artillery, Radar, Missile Systems) simulate weapon systems. The instantaneous state of the overall simulation is contained in a file known as the Master Electronic Warfare Environment (MEWE). Among other data, the MEWE contains position and mode of all pertinent EC entities.

In order to mimic the mode changes of real–world threat systems, the JARMSs monitor the environment via the MEWE and extract variables which are manipulated by a pre–determined set of rules. These rules, referred to as Tactics Algorithms (TAs), are tailored to the specific modes of each JARMS. A unique combination of tailored TAs define a specific JARMS behavior. The current EC simulation utilizes 28 TA templates.

The MEWE is scanned and TAs are processed in an iterative fashion to drive JARMS into different operational modes until termination of the engagement. Termination may occur due to range, target occult, break lock, or target kill.

Real world weapon systems may exhibit many distinct modes. However, JARMS have a maximum of eight, one inactive (off) and up to seven active. Observable characteristics of simulated modes are designed to correlate to selected modes of the real–world weapon system, to the limit of the fidelity of the system.These characteristics are made available, again via the MEWE, to the portion of the application that simulates the EC suite components onboard the ownships; i.e., radar warning receivers, electronic counter measure jammers, etc. These models attempt to detect and/or counter JARMS. This creates a closed–loop interaction between the JARMS and simulated EC suite components.

As shown (see Figure 2), TAs, and therefore mode changes, are a function of the following environmental inputs:

a. Ownship to JARMS range

b. Ownship altitude

c. Ownship elevation

d. Electronic Counter Measures (ECM) Effectivity

e. Weapon in Flight Status

f. Ownship to JARMS delta altitude (for Airborne Interceptors)

g. Ownship occult status

APPROACH

To provide a proof–of–concept that CI techniques are viable solutions to EC simulation, FS and NN "drop–in" replacements were designed that will respond to the same inputs and produce the same outputs as the current TAs, but with higher fidelity.

Envisioned advantages of a CI implementation include:

- New JARMS tactics are created and verified by a labor and time intensive trial and error approach. The trainability characteristic of NNs and the intuitive properties of FSs could reduce this effort.
- Processing of other information which is currently available, but not used, could be added to enhance the fidelity of the simulation. An example of such data is aspect of ownship to weapon relative bearing, an important real world parameter. Either a FS or NN implementation could process this data potentially creating a higher fidelity simulation than what currently exists.

To bound the problem, occult status was not used and only SAM type JARMS were investigated. Terrain masking data could be added later.

Due to sensitivity associated with EC applications, a fictitious SAM JARMS designated as the SA–A has

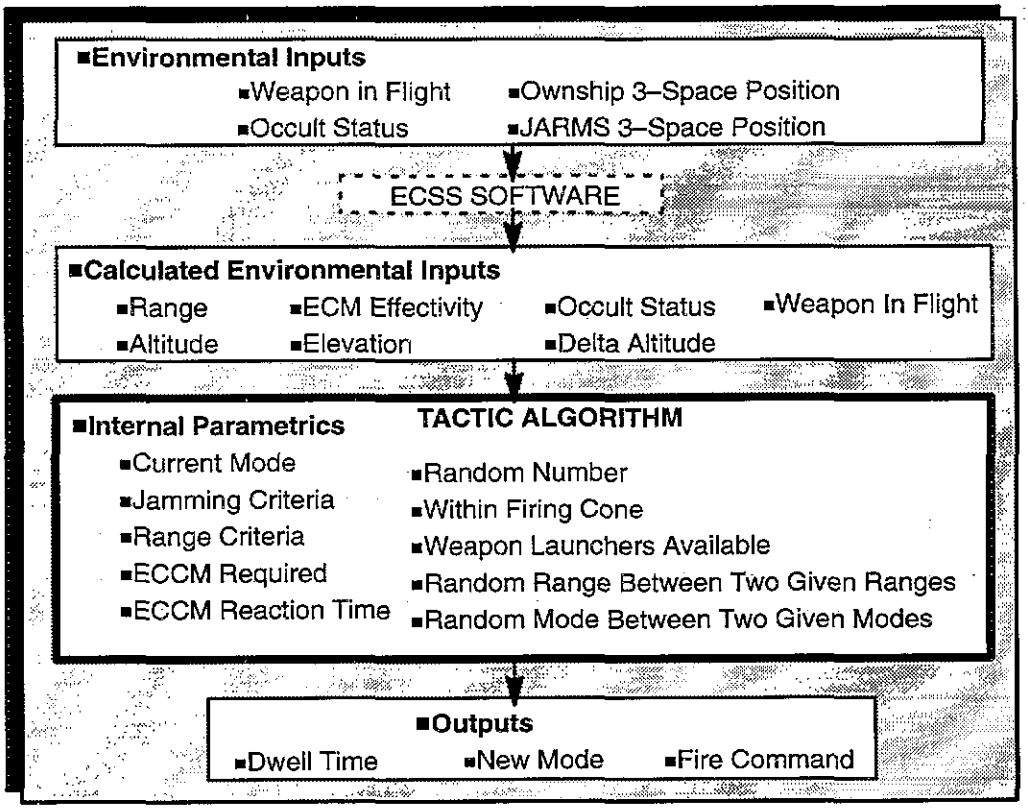

Figure 2. Existing Implementation Process Flow

been created. The SA–A utilizes parameters specified in the following TA definitions to model its behavior:

Current Mode of JARM = 0 (Off)
If Ownship Range > 90 nm
Then dwell in mode 1 for 60 sec
If Ownship Range < 90 nm
Then dwell in mode 2 for 30 sec

Current Mode of JARM = 1 (Search)
If Ownship Range > 65 nm
Then dwell in mode 2 for 60 sec
If Ownship Range < 65 nm
Then dwell in mode 1 for 30 sec

Current Mode of JARM = 2 (Acquisition)
If Ownship Range < 45 nm
If Jamming Effectivity > 40%
If Ownship Range > 35 nm
Then dwell in mode 3 for 20 sec
If Ownship Range < 35 nm
Then dwell in mode 4 for 20 sec
If Jamming Effectivity < 40%
Then dwell in mode 5 for 10 sec
If Ownship Range > 45 nm
Then dwell in mode 1 for 60 sec

Current Mode of JARM = 3 (Acquisition)
If ECCM is required
Then dwell in mode 4 for 20 sec
Else If Ownship Range > Previous Range
Then dwell in mode 1 for 60 sec

Current Mode of JARM = 4 (Low PRF)
If Ownship Range < 25 nm
If Jamming Effectivity > 50%
Then dwell in mode 3 for 20 sec
If Jamming Effectivity < 50%
Then dwell in mode 5 for 20 sec

Current Mode of JARM = 5 (High PRF)
If ECCM is required

Then dwell in mode 4 for 20 sec
If ECCM is not required
If Ownship Range > 25 nm
Then dwell in mode 3 for 20 sec
If Ownship Range < 27 nm
and Altitude > 500 ft
and Elevation < 70 deg
and Jamming Effectivity < 25%
Then Fire Weapon

Fuzzy System Solution

The Fuzzy Inference Process is the algorithmic process internal to the FS that provides the ability to manipulate abstract concepts in a fashion similar to human decision making. Fuzzy Systems need to interface with non–fuzzy applications, so the fuzzy process must accept and output crisp (mono–valued) data.

The Process is separated into three subfunctions:

1. "Fuzzification", where a crisp input value is transformed into a fuzzy value.

2. Rule Evaluation, where the fuzzy output truth values are computed.

3. "Defuzzification", where the computed fuzzy value is transformed into a crisp output value suitable for non–fuzzy applications.

Fuzzification is accomplished by applying crisp inputs to more than one function to produce a set of intermediate outputs. The shape and overlap of the functions in the domain of the range of the input provides ambiguity in the intermediate output set. For example, the FS range implementation for the SA–A used overlapping linear functions as shown (see Figure 3). The input data set is applied to previously developed rules, examples of which are shown (see Table 1).

The other input variables modeled in the same manner as the range include:

- Mode
- Jamming Effectivity
- ECCM Required
- Reaction Time
- Altitude
- Elevation

Each single input is transformed into one or more outputs. In the example provided, a range input of 45 nautical miles corresponds to the range membership function outputs of:

- 0 % Very Far
- 0 % Far
- 60 % MidNear
- 30 % Mid
- 0 % Near
- 0 % Very Near

A jamming effectivity of 53% corresponds to a jamming effectivity membership function output of:

- 0 % Very High
- 0 % High
- 100 % Mid
- 70 % Low Mid
- 0 % Low
- 0 % Very Low

During processing of the FS, all rules are exercised at the same time so, all these input variables are fuzzified as part of the system. An example of processing the two rules is given below:

1. As shown (see Figure 4), for each rule a product equivalent to the logical "and" of the inputs would be taken, in this case 60% for range and 100% for jamming effectivity. The logical "and" of these two inputs is the minimum, or 60%. It can be seen that the product of those sets will be zero for each input that results in a membership output of zero.

2. Next a product equivalent to the logical "or" of the resultant output of all the rules is taken. This corresponds to the superposition of the logically true portions of individual rules.

3. Outputs of these rules are input to an averaging function that uses centroid calculations on the resulting structures to yield crisp outputs. The crisp output values are input to the EC Simulation processes requiring control data of JARMS behavior.

The utility of the FS approach was immediately obvious. The FS used to model the SA–A was quick to construct, easy to debug and produced acceptable performance.

FS is implemented in ANSI C code that would be straightforward to add to the existing simulation.

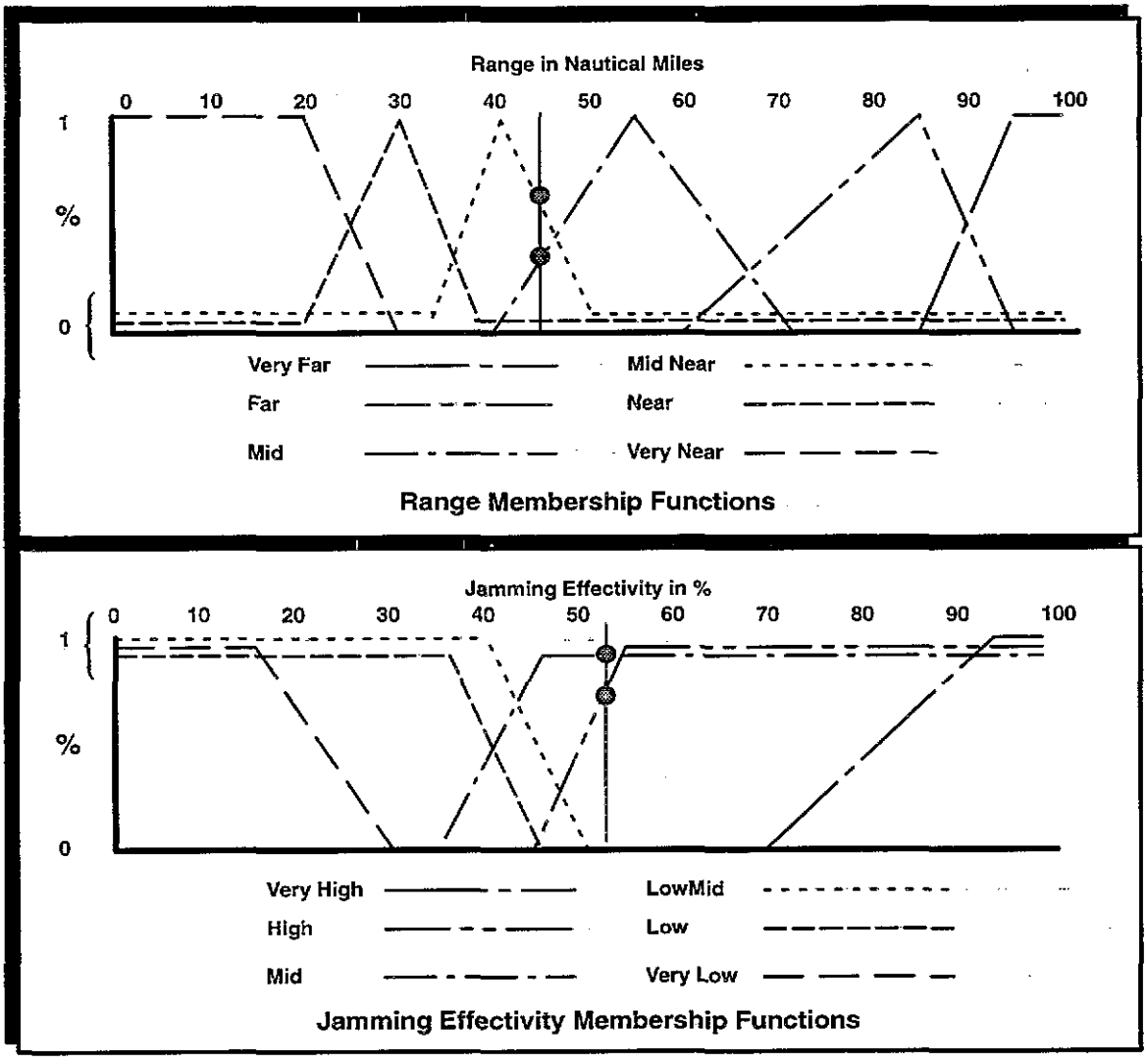

Figure 3. Membership Functions

There would be no hardware impact. The software impact would entail modifications to the executive/scheduler for inclusion of the new modules. The C source code would be cross complied on the development system and the resulting object code added to the simulation build.

Neural Network Solution

Neural nets provide a means of correlating data using simple processing elements. Basic neural net building blocks are neural units, analogous to the biological neuron. A neural unit accepts inputs and applies weights to them. These weights are the analog of biological synapses that connect neurons. Weighted inputs are summed and applied to a simple, predefined function. The result of this function is transmitted as the neural unit output. Units are associated into layers with the overall net being classifiable based on the number of layers.

Three layer neural nets have gained wide acceptance due to their versatility and ability to produce acceptable outputs with a minimum number of layers and their associated learning time overhead. These nets group units into an input layer, a hidden layer, and an output layer. The association of units in a three layer net is depicted (see Figure 5). This network represents the inputs to and outputs from one JARM. Each discrete input is associated with an input node, and each discrete output is associated with an output node. The number of nodes in the hidden layer are directly

1. if range is MidNear and jam_eff is Mid then new_mode is Acq_ECCM
2. if range is Very Near and jam_eff is Very Low then new_mode is High_PRF
1. if current_mode is Activate and range is VeryFar then new_mode is Search
2. if current_mode is Activate and range is Far then new_mode is Acquisition
3. if current_mode is Search and range is Far then new_mode is Search
4. if current_mode is Search and range is Mid then new_mode is Acquisition
5. if current_mode is Acquisition and range is MidNear and jam_eff is Mid and range is Near then new_mode is Low_PRF
6. if current_mode is Acquisition and range is MidNear and jam_eff is Low then new_mode is High_PRF

Table 1. Samples of FS Rules

proportional to the number of cases the neural net must learn and desired precision of output data.

NNs form the transfer functions that map input data to output data. Neural nets are trained, not programmed. The training process provides inputs to the net for which there exist known outputs. The collection of weighting factors (sometimes referred to as the gain matrix) between the elements of the net are then iteratively perturbed via the back–propagation algorithm so as to drive the error in the output to a predetermined value. Once trained, a network is ready to be tested by accepting non–training data. For each new input case, the network produces a best estimate output with a corresponding measure of uncertainty.

Speed is a primary advantage over conventional processing. Neural nets make connections almost instantaneously and processes data in parallel. Also NNs can produce answers based on incomplete input data sets.

Upon examination of this application and the TAs, several NN implementations were investigated. In the first implementation, a NN configuration was developed that attempted to accommodate all SAM type JARMS. This implementation used a NN for each JARMS TA/mode.

The goal of the first implementation was to develop a set of NNs consisting of one NN for each of eight TAs used. All eight NNs used the same input and output nodal configurations. However, these nets were trained with different case sets of data. The unique case sets were developed to support each TA's behavior the network was to learn. This approach could then permit an operator to develop new SAM JARMS neural nets by simply creating appropriate training and test case sets.

The resulting NN consisted of 6 inputs and 3 outputs as shown below.

Inputs:

- JARM Identification Code
- Current Mode
- Range
- Jamming Effectivity
- Altitude
- Elevation

Outputs:

- Next Mode
- Dwell Time
- Fire Weapon Command

A neural network of this configuration was then constructed for the 6 TAs of the SA–A. Training cases were developed and used to teach the networks the SA–A's behavior. These networks

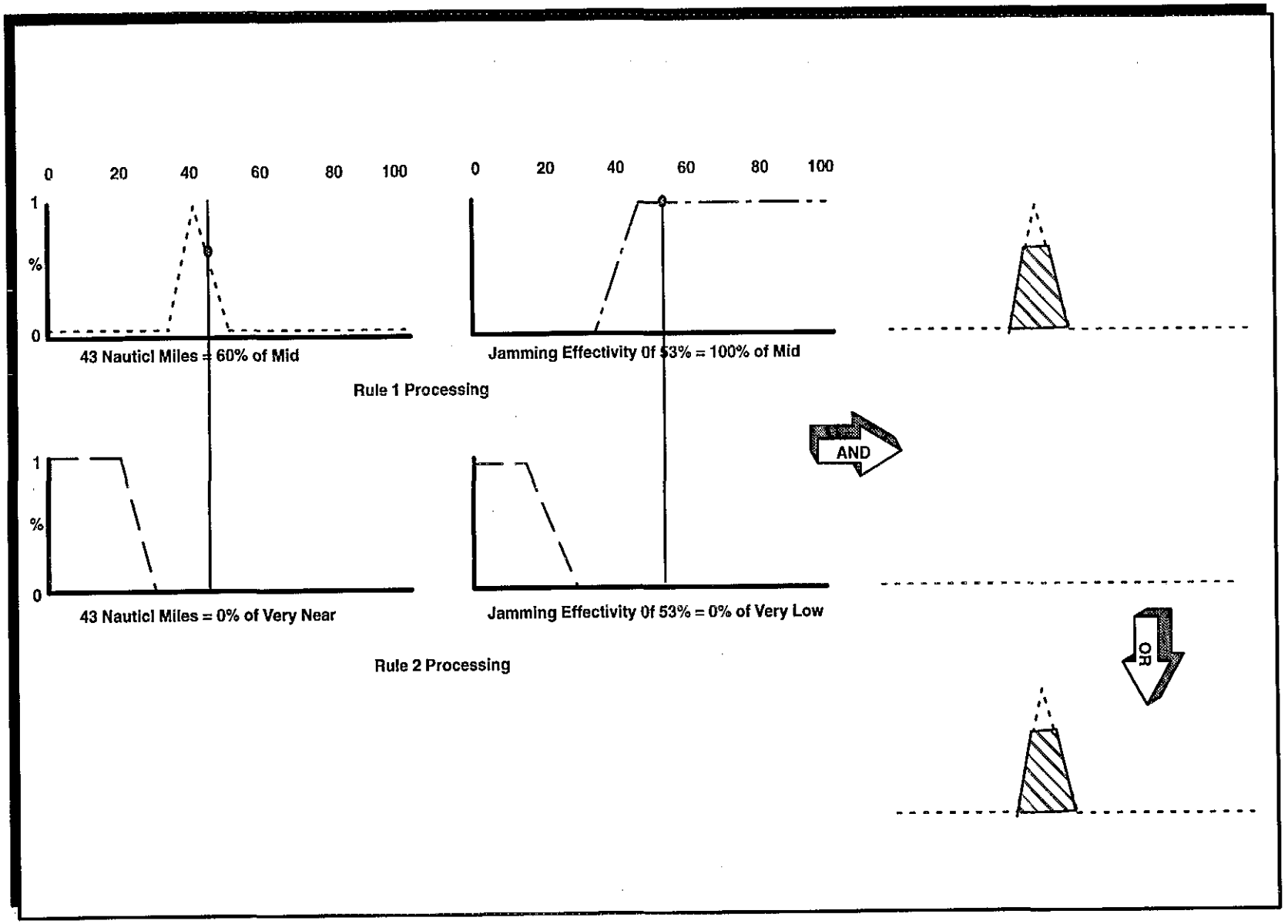

Figure 4. Rule Processing

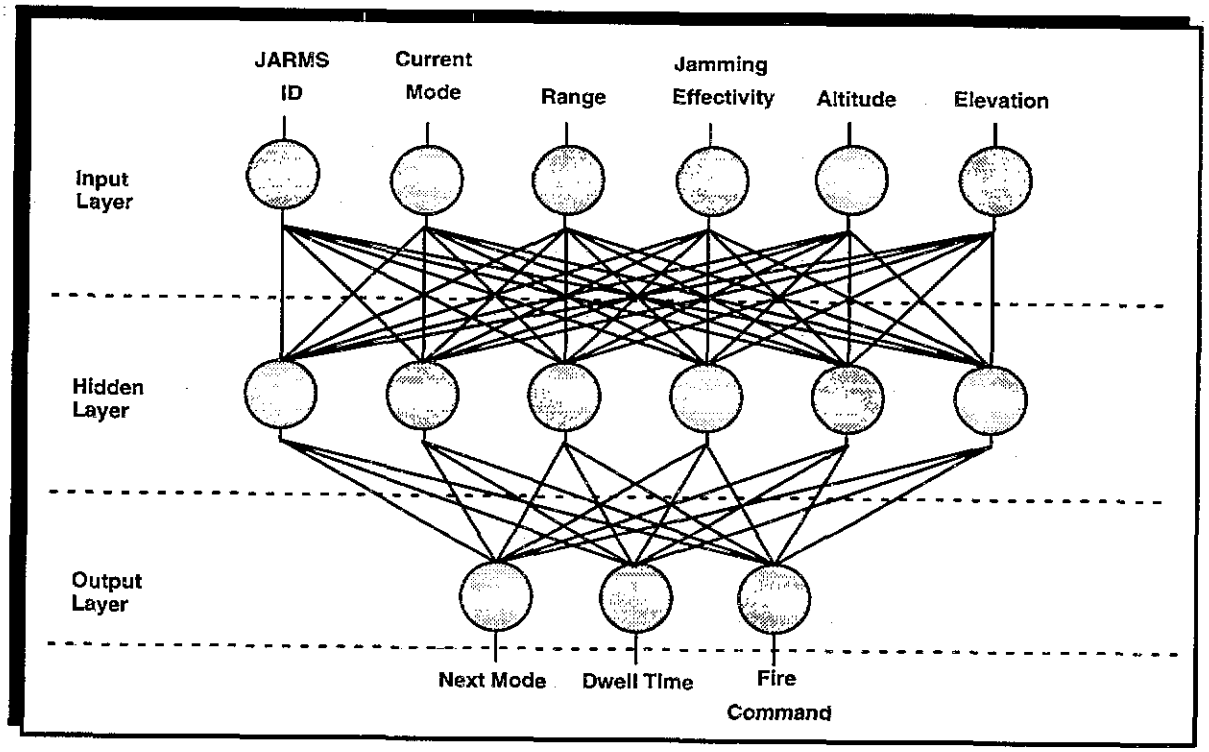

Figure 5. SAM Type JARMS
Three Layer Neural Network Configuration

were then tested using different data than that used for training.

For example, the network developed to model the particular TA behavior used in mode 0 was trained on inputs varying range from 40 to 160 nautical miles. It was trained to go to mode 2 for a dwell time in that mode of 30 seconds if the range is 89 nautical miles or less. If the range is 91 nautical miles or more, the next mode is 1 for a dwell time of 60 seconds in that mode.

When tested with the discrete range value of 72 nautical miles, the network yields the next mode value of 2.25 and a dwell time of 18.63 seconds. The EC system operates in distinct modes, the next mode value can be truncated to its integer value of mode 2. However, the non-integer 18.63 second dwell time can be used instead of the 20 seconds used in the existing if–then–else logic.

As can be deduced from examination, each of the SA–A TAs is not a function of all the possible inputs. Thus, a second approach was considered wherein each TA was individually modeled. This approach, like the first, yielded six networks to support modeling the SA–A. This implementation of the TAs restricted neural net inputs and outputs to only those required to capture the behavior of an individual TA. For example, inputs for the TA used in SA–A mode 3 are current mode, range, and jamming effectivity. Outputs are next mode and dwell time in the next mode.

Again, cases were developed and neural nets trained. One test case constructed for the TA used in mode 3 consists of a range input of 33 nautical miles and jamming effectivity input of 43%. For this case, the NN yielded a next mode value of 3.87 with a dwell time in that mode of 20.07 seconds. The next mode interpretation could be truncated to a mode 3 and the dwell time duration as the value concluded by the network.

The latter networks proved easier to develop than those configured in the former approach. This is due to the manner in which the problem was

decomposed and limits on the number of cases that had to be learned by the networks.

CONCLUSIONS

This paper has discussed two CI methods for replacement of the current EC Simulation TA processing: Neural Networks and Fuzzy Systems. It has been shown that FSs appear to be superior to NNs for this application. The NN solution appears to be less suited for the following reasons:

- The NN solution pursued for this application is open–ended. Decomposition of the data to an optimum NN implementation was not obtained during the course of this study.
- The complexity of the problem did not allow a NN or collection of NNs to be created which produced improved performance of the TAs modelled.
- The training time of the NN was not less, and in fact much longer, than the existing method to produce new JARMS.

The FS approach appears to be stronger since TAs are more a collection of rules than a collection of data, and that for this application a knowledge based approach such as a FS is superior to the data driven solution provided by neural networks.

Utility of the FS approach was immediately obvious during this study. The FS tool permitted quick graphical construction of the SA–A TAs modelled, easy debugging, generated commented code, and produced acceptable models. However, a much greater effort than that conducted in this study will be required to produce a useful product to integrate into the existing EC Simulation.

FSs could provide an improved performance of the EC Simulation. This is due to their capability of expanding knowledge of the if–the–else logic and using computational techniques to extract better conclusions.

The FS developed is implemented in ANSI C code, which is compatible with the existing simulation. Changes would entail modifications including the new FS modules and deletion of the existing TA if–the–else structure code. The FS C source code would be cross complied on a development system and resulting object code added to the EC Simulation build.

BIBLIOGRAPHY

1. How Neural Networks Learn from Experience
   Geoffrey E. Hinton
   Scientific American, September 1992

2. Neural Networks
   Peter J. Denning
   American Scientist, September–October 1992

3. Working with Neural Nets
   Dan Hammerstrom
   IEEE Spectrum July 1993

4. Fuzzy Logic – From Concept to Implementation
   Aptronix, Inc.