Creating Fuzzy Decision Algorithms Using Genetic Program Based Data Mining

J. F. Smith III
Code 5741
Naval Research Laboratory
Washington, DC, 20375-5320
james.smith@nrl.navy.mil

T. H. Nguyen
Code 5741
Naval Research Laboratory
Washington, DC, 20375-5320
thanhvu.nguyen@nrl.navy.mil

Abstract **- A data mining procedure for automatic determination of fuzzy decision tree structure using a genetic program is discussed. A genetic program is an algorithm that evolves other algorithms or mathematical expressions. Methods of accelerating convergence of the data mining procedure including a new innovation based on computer algebra are examined. Experimental results related to using computer algebra are given. Comparisons between trees created using a genetic program and those constructed solely by interviewing experts are made. A genetic program evolved tree is shown to be superior to one created by hand using expertise alone. Finally, additional methods that have been used to validate the data mining algorithm are discussed.**

I. INTRODUCTION

Two fuzzy logic based resource managers (RMs) have been developed that automatically allocate resources in real-time. Both RMs were evolved by genetic programs (GPs). The GPs were used as data mining functions. Both RMs have been subjected to a significant number of verification experiments.

The first to be discussed is an RM that allocates electronic (EA) resources in real-time over very general platforms [1]. This RM is referred to as the EARM. The EARM is designed to work in many different environments and to be effective against very general enemies.

The second RM that will be considered automatically allocates unmanned aerial vehicles (UAVs) that will ultimately measure atmospheric properties in a cooperative fashion without human intervention. This RM will be referred to as the UAVRM. It consists of a pre-mission planning algorithm and a real-time control algorithm that runs on each UAV during the mission allowing the UAVs to automatically cooperate.

Section II provides an overview of the two RMs to be considered. Section III describes the creation of the EARM's isolated platform decision tree using a GP as a data mining function. Section IV discusses methods of validating data mining results for the EARM. Section V examines how a fuzzy decision tree for the UAVRM was created through GP based data mining. Section VI considers the validation of the data mining results described in section V. Finally, section VII provides a summary.

II. Overview of the Resource Managers

The particular approach to fuzzy logic used by the RMs is the fuzzy decision tree [2, 3]. The fuzzy decision tree is an extension of the classical artificial intelligence concept of decision trees. The nodes of the tree of degree one, the leaf nodes, are labeled with what are referred to as root concepts. Nodes of degree greater than unity are labeled with composite concepts, i.e., concepts constructed from the root concepts [4,5] using logical connectives and modifiers. Each root concept has a fuzzy membership function assigned to it. Each root concept membership function has parameters to be determined. For the EARM the parameters were determined by genetic algorithm based optimization [5, 6]. For the UAVRM the parameters were set based on expertise.

The EARM resource manager is made up of four decision trees, the isolated platform decision tree (IPDT), the multi-platform decision tree (MPDT), the fuzzy parameter selection tree and the fuzzy strategy tree. The IPDT provides a fuzzy decision tree that allows an individual agent to respond to a threat [1, 5]. The other decision trees are discussed in the literature.

The UAVRM consists of three fuzzy decision trees. Only the creation of the AUP tree by a GP will be considered in this paper. The AUP tree gets its name from its function, which is to "Assign UAVs to Paths" (AUP). The AUP tree makes use of the Risk tree which is discussed in the literature [7].

To be consistent with terminology used in artificial intelligence and complexity theory [8], the term "agent" will sometimes be used to mean platform. Finally, the terms "blue" and "red" will refer to "agents" on opposite sides of a conflict, i.e., the blue side and the red side.

III. DISCOVERING THE IPTD FUZZY DECISION TREE'S STRUCTURE USING A GENETIC PROGRAM

The IPDT allows a blue agent that is alone or isolated to determine the intent of a detected agent. It does this by processing data measured by the sensors. Even when an incoming agent's ID is very uncertain, the IPDT can still establish intent based on kinematics. When faced with multiple incoming agents the IPDT can establish a queue of which agents to attack first. Various subtrees of the IPDT have been discussed extensively in the past [1, 5].

Data mining is the efficient extraction of valuable non-obvious information embedded in a large quantity of data [9]. Data mining consists of three steps: the construction of a database that represents truth; the calling of the data mining function to extract the valuable information, e.g., a clustering algorithm, neural net, genetic algorithm, genetic program, etc; and

1-4244-0363-4/06/$20.00

finally determining the value of the information extracted in the second step, this generally involves visualization.

In a previous paper a genetic algorithm (GA) was used as a data mining function to determine parameters for fuzzy membership functions [5]. Here, a different data mining function, a genetic program (GP) [10] is used. A genetic program is a problem independent method for automatically evolving computer programs or mathematical expressions.

The GP data mines fuzzy decision tree structure, i.e., how vertices and edges are connected and labeled in a fuzzy decision tree. The GP mines the information from a database consisting of military scenarios. Whereas, the GA based data mining procedures determine the parameters of and hence the form of fuzzy membership functions, the GP based procedure actually data mines fuzzy if-then rules. GP based data mining will be applied to the construction of the IPDT.

To use the genetic program it is necessary to construct terminal and function sets relevant to the problem. The terminal set used for construction of subtrees of the IPDT is given below.

T={close, heading_in, elevation, ranging, banking, elevating, interaction, friend, lethal, uncertain, marginal-ID}. (1)

The elements of this terminal set are fuzzy root concepts that are explained in the literature [5].

The function set, F, consists of the logical operations of "AND" and "OR" as well as the logical modifier "NOT," i.e.,

F={AND1, OR1, AND2, OR2, NOT}. (2)

More than one form of AND and OR appear in (2), i.e., AND1, AND2, OR1, OR2, etc., because fuzzy logic allows more than one mathematical form for AND and OR.

Let A and B represent fuzzy membership functions then elements of the function set are defined as

$$AND1(A,B) = min(A,B); \tag{3}$$

$$OR1(A,B) = max(A,B); \tag{4}$$

$$AND2(A,B) = A \cdot B; \tag{5}$$

$$OR2(A,B) = A + B - A \cdot B; \tag{6}$$

and

$$NOT(A) = 1 - A. \tag{7}$$

The fitness function for data mining the IPDT subtree is

$$fitness\ (i) \equiv g(i, n_{db}, n_{time}, \tau) - \alpha \cdot l(i) \tag{8}$$

with

$$g(i,n_{db},n_{time},\tau) \equiv$$

$$\frac{1}{n_{time} \cdot n_{db}} \sum_{j=1}^{n_{db}} \sum_{k=1}^{n_{time}} \chi(\tau - |\mu_{gp}(i,t_k,e_j) - \mu_{expert}(t_k,e_j)|) \tag{9}$$

where the function g is the basic fitness, e_j is the j^{th} element of the database; t_k is the k^{th} time step; n_{db} is the number of elements in the database; n_{time} is the number of time steps; τ is the tolerance; $\mu_{gp}(i,t_k,e_j)$ is the output of the fuzzy decision tree created by the GP for the i^{th} element of the population for time step t_k and database element e_j; $\mu_{expert}(t_k,e_j)$ is an expert's estimate as to what the fuzzy decision tree should yield as output for time step t_k and database element e_j; α is the parsimony coefficient; $l(i)$ is the length of the i^{th} element of the population, i.e., the number of nodes in the fuzzy decision tree corresponding to the i^{th} element; and $\chi(t)$ is the Heaviside step function, which is unity for $t \geq 0$ and zero otherwise.

Observe, that the form of (8, 9) reflects that the expert's estimate, $\mu_{expert}(t_k,e_j)$ is uncertain, and need only be reproduced within a tolerance, τ. Also, to increase the robustness of the GP created tree, the fitness of the fuzzy decision tree used by the GP is found by averaging over time and the database.

The parsimony pressure, $\alpha \bullet l(i)$, appearing on the right-hand-side of (9) provides a penalty that reduces the i^{th} population element's fitness if it is longer than needed. Thus given two trees that are both effective, the smaller tree will have the higher fitness. This provides a computational implementation of Occam's razor [11].

It is observed when using GPs that candidate solutions increase significantly in length during the evolutionary process. It is an empirically observed rule [12] that for every 50 generations, trees will grow by a factor of three in length. Many adhoc procedures have been developed to control this aspect [12-13], e.g., parsimony pressure described above, Koza depth limits, tournaments, etc. These procedures have the problem that they can prune away parts of a decision tree useful during low probability events.

When mathematical expressions are constructed by a GP that reproduce the entries in a database within some tolerance, the process is referred to as symbolic regression [12]. It is found in symbolic regression that candidate solutions are frequently not in algebraic simplest form and this is the major source of their excess length. When candidate solutions are too long this is referred to as bloat.

A simple method of reducing length of candidate solutions that are mathematical expressions is to introduce computer algebra for automated simplification. This results in not only simpler solutions, but also the GP converges more rapidly and its CPU time requirements are reduced. This is described in greater detail in section V.H.

The GPs applied in both the RM examples of this paper use the following three common termination criteria: the fitness has reached a certain value; the fitness has not improved in a certain number of generations, or a maximum number of generations has been reached.

Fig. 1 depicts the IPDT subtree evolved by using GP based data mining. This subtree was originally drawn based on experts' intuition [5,13]. The fact that the GP was able to evolve a tree already known based on interviewing experts is a form of validation for the GP based procedure.

In Fig. 1, the mathematical forms of the logical connectives label each vertex and a circle on an edge denotes the logical modifier, "NOT." The fuzzy concepts, labeling each box, are described in detail in the literature [5,13].

This subtree of the IPDT has been rediscovered by data mining a database of military scenarios using a GP. Other more sophisticated trees have been discovered by GP based data mining, but this simple tree is considered here to illustrate the process. The GP in many different runs was successful in constructing this subtree as expected, however, it did not always construct the same tree. The concept on the right-hand-side of the tree labeled "status" is a placeholder. In some trees constructed by the GP, its value was "not a friend" in others "status" took the value "lethal." The value "lethal" was originally proposed by experts, but has proven to be less effective than the GP suggested value of "not a friend."

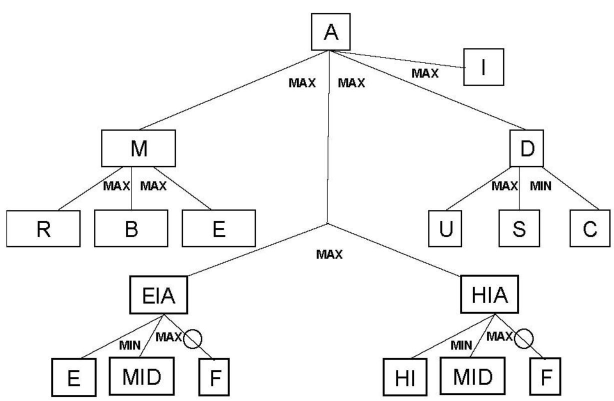

Figure 1: The IPDT where A="attacking," I="interaction," D="dangerous," U="uncertain," S="status," C="close," HIA="heading-in-attack," HI="heading-in," MID="marginal-ID," F="friend," E="elevation," EIA="elevated attack."

IV. Evaluation of the Data Mined IPDT Subtree

Three different approaches for the validation of the data mined decision trees are discussed in this section. These approaches are the evaluation of the EARM within a digital game environment [5,13]; testing the EARM using a hardware simulator; and comparing the decision trees obtained through data mining to similar ones created solely through interviewing experts.

The scenario generator (SG) game simulation is described in detail elsewhere [5,13]. So only a quick summary will be given here. The SG allows the creation of a very general digital simulation environment that may have a map with desert, forest, jungle, urban areas, and water. Very general blue agents, i.e., the defending platforms each one of which runs its own copy of the EARM can be placed within the environment. The agents can be ships, planes, helicopters, soldiers, decoys, etc. The SG allows the agents to be equipped with very general sensors, weapons, etc. Likewise, the SG allows very general red agents to be created and well equipped. The SG has two modes of operation, computer vs. computer (CVC) mode, and human vs. computer mode (HVC). In both modes each blue agent has its own copy of the EARM. The blue agent's EARM exercises all control functions over that agent and only that agent. In CVC mode each red agent is controlled by its own computerized logic different from the EARM. In HVC mode, a human expert controls one red agent per time step through a GUI. The human player can select a different red agent each time step to control. Those red agents not under human control run under computer logic as in CVC mode. Many different conflicts can be simulated using the SG, the results are stored in a database and also a computer movie. Human experts have evaluated many of the computer movies and agreed on the excellent decisions made by the decision tree of Fig. 1.

Evaluation using a hardware simulator (HS) is similar to the work done with the SG, but in this case the digitally simulated radars, communication systems, and sensor displays of the SG are replaced with real hardware systems. In this application the RM is used as a controller on the HS, allowing evaluation of the RM. As in the previous approach to validation, experts concluded that the decision tree of Fig. 1 made excellent decisions during the HS test.

The final contribution to the validation effort consists of comparing decision trees created through data mining to those designed solely using rules obtained by interviewing experts. As discussed in the previous section the GP is able to recreate the IPDT subtree found through consulting with experts to acquire "if-then" rules. However, as stated above, the GP does not always reproduce the known tree. In the case of the IPDT, the second tree created through data mining is arguably superior to the one arising from "if-then" rules provided by experts. It is useful to be able to recreate known decision trees, this establishes confidence in the data mining process. The most important ability of the GP based data mining procedure is to be able to construct decision trees for situations for which human expertise is not available.

V. GP Creation of the AUP Tree

The previous two sections concentrated on how the GP was used to evolve a subtree of the IPDT of the EARM. This section will emphasize using the GP as a data mining function to automatically create the AUP tree of the UAVRM.

A. Motivation for the AUP Tree

Knowledge of meteorological properties is fundamental to many decision processes. The UAVRM enables a team of UAVs to cooperate and support each other as they measure atmospheric meteorological properties in real-time. Each UAV has onboard its own fuzzy logic based real-time control algorithm. The control algorithm renders each UAV fully autonomous; no human intervention is necessary. The control algorithm aboard each UAV will allow it to determine its own course, change course to avoid danger, sample phenomena of interest that were not preplanned, and cooperate with other UAVs.

The UAVRM determines the minimum number of UAVs required for the sampling mission. It also determines which points are to be sampled and which UAVs will do the sampling. To do this, both in the planning and control stages it must solve an optimization problem to determine the various paths that must be flown. Once these paths are determined the UAVRM uses the AUP fuzzy decision tree to assign UAVs to the paths.

B. Structure of the AUP Tree and Its Subtrees

The AUP fuzzy decision tree is displayed in Fig. 2. The various fuzzy root concepts make up the leaves of the tree, i.e., those vertices of degree one. The vertices of degree higher than one are composite concepts.

Starting from the bottom left of Fig. 2 and moving to the right, the fuzzy concepts “risk-tol,” “value”, “fast,” and “low risk,” are encountered. The fuzzy concept “risk-tol” refers to an individual UAV’s risk tolerance. This is a number assigned by an expert indicating the degree of risk the UAV may tolerate. A low value near zero implies little risk tolerance, whereas, a high value near one implies the UAV can be subjected to significant risk.

The concept “value” is a number between zero and one indicating the relative value of a UAV as measured against the other UAVs flying the mission. The concept “value” changes from mission to mission depending on which UAVs are flying.

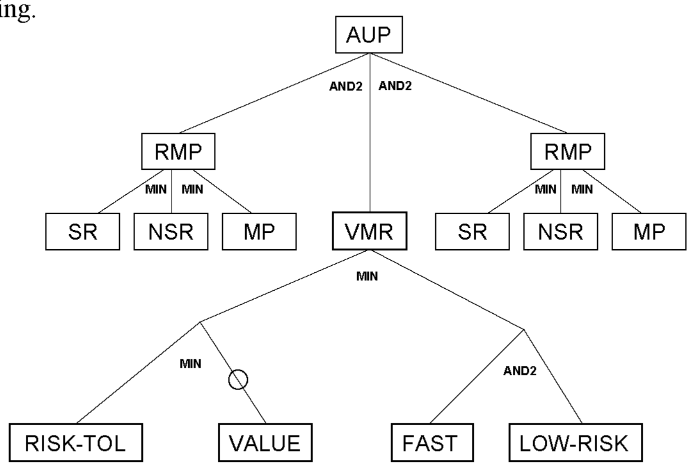

Figure 2: The AUP subtree for the UAVRM.

The concept “fast” relates to how fast the UAV is and builds in measures of the UAV’s reliability estimates as well as its risk tolerance and the mission’s priority.

The rightmost concept is “low risk.” It quantifies experts’ opinions about how risky the mission is. It takes a value of one for low risk missions and a value near zero for high risk missions.

These four fuzzy root concepts are combined through logical connectives to give the composite concept “VMR.” Each vertex of the “VMR” tree uses a form of “AND” as a logical connective. In fuzzy logic, logical connectives can have more than one mathematical form. Based on expertise it was useful to allow two types of ANDs to be used. The two mathematical forms of AND used are the “min” operator and the algebraic product denoted in Fig. 2 as “AND2.” When a “min” appears on a vertex then the resulting composite concept arises from taking the minimum between the two root concepts connected by the “min.” When an “AND2” appears it means that the resulting composite concept is the product of the fuzzy membership functions for the two concepts connected by the AND2.

The final subtree of AUP that needs to be described is RMP. The RMP tree appears twice on the AUP tree. RMP consists of a “min” operation between three fuzzy concepts. These concepts are “sr” which refers to an expert’s estimate of the sensor reliability, “nsr” which refers to an expert’s estimate of the non-sensor system reliability and “MP” a fuzzy concept expressing the mission’s priority.

The AUP tree is observed to consist of the VMR subtree and two copies of the RMP subtree with AND2 logical connectives at each vertex. These fuzzy concepts and their related fuzzy membership functions are explained in much greater detail in [7]. Additional explanations and motivation for the work can be found in [14].

The next few subsections will develop the terminal set, function set, and fitness functions necessary for the GP to be used as a data mining function to automatically create the AUP tree.

C. The GP’s Terminal Set and Function Set for Creating AUP

The terminal set used to evolve the AUP consisted of the root concepts from the AUP tree and their complements, i.e., the terminal set T is given by

T={risk-tol, value, fast, low-risk, sr, nsr, MP, not-risk-tol, not-valuable, not-fast, not-low-risk, not-sr, not-nsr, not-MP}. (10)

Let the corresponding fuzzy membership functions be denoted as

$$\begin{aligned}&\{\mu_{risk-tol}, \mu_{value}, \mu_{fast}, \mu_{low-risk}, \mu_{sr}, \mu_{nsr}, \dots \\ &\mu_{MP}, \mu_{not-risk-tol}, \mu_{not-valuable}, \mu_{not-fast}, \dots \\ &\mu_{not-low-risk}, \mu_{not-sr}, \mu_{not-nsr}, \mu_{not-MP}\}\end{aligned} \tag{11}$$

By including in the terminal set a terminal and its complement, e.g., "risk-tol," and "not-risk-tol"; "value" and "not-valuable"; etc., it is found that bloat is less and convergence of the GP is accelerated. This is a recent innovation which was not used when the IPDT was evolved using a GP. Additional bloat control procedures are described below.

The mathematical form of the complement whether it appears in the terminal set or is prefixed with a "NOT" logical modifier from the function set is one minus the membership function. To make this more explicit

$$\mu_{NOT(A)} = \mu_{not-A} = 1 - \mu_A \tag{12}$$

where *NOT(A)* refers to the application of the logical modifier *NOT* from the function set to the fuzzy concept A from the terminal set. The notation, *not-A* refers to the terminal which is the complement of the terminal A.

The function set, denoted as F, consists of

$$F=\{AND1, OR1, AND2, OR2, NOT\} \tag{13}$$

where the elements of (13) are defined in (3-7).

D. The Database to be Data Mined

The database to be data mined is a scenario database like that described for the evolution of the IPDT. In this instance scenarios are characterized by values of the fuzzy membership functions for the elements of the terminal set plus a number from zero to one indicating the expert's opinion about the value of the fuzzy membership function for AUP for that scenario.

E. The Fitness Function

The input-output fitness for mining the scenario database takes the form

$$f_{IO}(i, n_{db}) \equiv \frac{1}{1 + 2 \cdot \sum_{j=1}^{n_{db}} \left| \mu_{gp}(i, e_j) - \mu_{expert}(e_j) \right|}. \tag{14}$$

where e_j is the j^{th} element of the database; n_{db} is the number of elements in the database; $\mu_{gp}(e_j)$ is the output of the fuzzy decision tree created by the GP for the i^{th} element of the population for database element e_j; and $\mu_{expert}(e_j)$ is an expert's estimate as to what the fuzzy decision tree should yield as output for database element e_j.

The AUP tree is evolved in three steps. The first step involves evolving the VMR subtree; the second step, the RMP subtree and the final step, the full AUP tree. In the second and third steps, i.e., evolving the RMP subtree and full AUP tree from the RMP and VMR subtrees, only the input-output (IO) fitness in (14) is calculated.

When evolving the VMR subtree a rule-fitness is calculated for each candidate solution. Only when the candidate's rule fitness is sufficiently high is its input-output fitness calculated. The use of the rule-fitness helps guide the GP toward a solution that will be consistent with expert rules. Also the use of the rule fitness reduces the number of times the IO fitness is calculated reducing the run time of the GP.

F. Rule Fitness and Fuzzy Rules to Accelerate Convergence

After some preliminary definitions of crisp and fuzzy relations, a set of crisp and fuzzy rules that were used to help accelerate the GP's creation of the VMR subtree are given.

Let T be a fuzzy decision tree that represents a version of the VMR subtree, that is to be evolved by a genetic program (GP). Let A and B be fuzzy concepts. Then let $\gamma_{share}(T, A, B) = 1$ if A and B share a logical connective denoted as C and $\gamma_{share}(T, A, B) = 0$, otherwise.

Furthermore, define the fuzzy relation

$$\mu_{com}(T, A, B, C) = \begin{cases} 0.4 & if \quad C = AND1 \quad or \quad AND2 \\ 0.1 & if \quad C = OR1 \quad or \quad OR2 \\ 0, & otherwise \end{cases} \tag{15}$$

The following rules were used to accelerate the GP's convergence and to help produce a result consistent with human expertise.

R1. "not-valuable" and "risk-tol" must share a logical connective, denoted as C_1, i.e., it is desired that $\gamma_{share}(T, not-valuable, risk-tol) = 1$

R2. "not-valuable" and "risk-tol" strongly influence each other, so they should be connected by AND1 or AND2. So it is desired that $\mu_{com}(T, not-valuable, risk-tol, C_1) = .4$

R3. "fast" and "low-risk" have an affinity for each other. They should share a logical connective, denoted as C_2, i.e., it is desired that $\gamma_{share}(T, fast, low-risk) = 1$

R4. The fuzzy root concepts "fast" and "low-risk" strongly influence each other, so they should be connected by AND1 or AND2. So it is desired that $\mu_{com}(T, fast, low-risk, C_2) = .4$.

R5. There is an affinity between the fuzzy root concepts $C_1(not-valuable, risk-tol)$ and $C_2(fast, low-risk)$, they are connected by a logical connective denoted as C_3, i.e., it is desired that,

$$\gamma_{share}(T, C_1(not-valuable, risk-tol), C_2(fast, low-risk)) = 1$$

R6. The fuzzy composite concepts $C_1(not-valuable, risk-tol)$ and $C_2(fast, low-risk)$ strongly influence each other so it is desired that

$$\mu_{com}(T, C_1(not-valuable, risk-tol), C_2(fast, low-risk), C_3) = .4$$

R7. The elements of $D=\{not\text{-}valuable,\ risk\text{-}tol,\ fast,\ low\text{-}risk\}$ should appear on the tree T at least once, i.e.,

$$\mu_{4C}(T) = \begin{cases} 1 & if \quad D's \quad elements \quad present \\ 0 & otherwise \end{cases}$$

R8. The elements of D should probably appear only once.

$$\mu_{4C1T}(T) = \begin{cases} .6, & if \quad appear \quad only \quad once \\ .2, & if \quad any \quad appear \quad twice \\ .1, & if \quad any \quad appear \quad \geq 2 \quad times \\ 0, & otherwise \end{cases}$$

The rule-fitness (RF), denoted as $\mu_{RF}(T)$ is defined to be

$$\begin{aligned} \mu_{RF}(T) \equiv \frac{1}{8} \cdot [(\gamma_{share}(T, not-valuable, risk-tol) + \\ \mu_{com}(T, not-valuable, risk-tol, C_1) + \\ \gamma_{share}(T, fast, low-risk) + \\ \mu_{com}(T, fast, low-risk, C_2) + \\ \gamma_{share}(T, C_1(not-valuable, risk-tol), C_2(fast, low-risk)) + \\ \mu_{com}(T, C_1(not-valuable, risk-tol), \\ C_2(fast, low-risk), C_3) + \mu_{4C}(T) + \mu_{4C1T}(T)] \ . \end{aligned}$$

G. Tournament Selection

The GP program uses tournament selection [12] to accelerate convergence and as one method of dealing with bloat control. In tournament selection, the population is partitioned into tournament subpopulations (TPs). For each TP, the subset of maximum fitness chromosomes (SMFC) is found. If the SMFC has one element then that chromosome is the winner of the tournament for that TP. If SMFC has more than one element then the subset of minimum depth chromosomes (SMDC) is selected from SMFC. If SMDC has only one element then it is the winner of the tournament for that TP, otherwise a chromosome is selected from the SMDC at random to be the TP's winner.

H. Computer Algebra

In the preceding sections bloat has been controlled using adhoc procedures based on tree depth and parsimony pressure. Most of the bloat in evolving mathematical expressions with a GP arises from the expressions not being in algebraic simplest form [12]. With that observation in mind computer algebra routines have been introduced that allow the GP to simplify expressions. The following is a partial list of algebraic simplification techniques used during the evolution of the IPDT and the AUP tree. The simplification routines used when evolving AUP are more sophisticated than those employed for the creation of the IPDT.

One routine simplifies expressions of the form $NOT(NOT(A)) = A$. This can be more complicated than it initially appears, since the NOT logical modifiers can be separated on the fuzzy decision tree.

Another simplification procedure consists of eliminating redundant terminals connected by an AND1 logical connective. An example of this is $AND1(A,A) = A$.

Like the case with the logical modifier NOT there can be a separation between the AND1s and the terminals that add complexity to the simplification operation.

The third algebraic simplification example is like the second. It involves simplifying terminals connected by OR1s. Like AND1, separation between terminals and OR1 can increase the complexity of the operation.

VI. Evaluation of the AUP Tree

The AUP tree has been the subject of many experimental tests and has been very successful in producing expected results. Some of these tests are described in [7]. Additional tests are described in [14]. The test in [14] were actually conducted using an AUP decision rule as opposed to the GP evolved decision tree. The decision rule was constructed by hand using human expertise. It is possible through a mathematical transformation to obtain the AUP decision tree from the AUP decision rule. When the AUP decision rule or decision tree are applied to the same experiments, the expected results are obtained. Finally, both the AUP decision rule and decision tree have shown excellent performance in all experiments conducted to date.

VII. Summary

A genetic program (GP) has been used as a data mining (DM) function to automatically create decision logic for two different resource managers (RMs). One RM referred to as the EARM, automatically allocates electronic attack (EA) resources distributed over different platforms. The second RM, referred to as the UAVRM, automatically controls a group of unmanned aerial vehicles (UAVs) that are cooperatively making atmospheric measurements.

DM procedures that use a GP as a data mining function are described for both RMs. The resulting decision logic for the RMs is rendered in the form of fuzzy decision trees. The different fitness functions, bloat control methods, data bases, etc., for the two RMs are described. Innovative bloat control methods using computer algebra based simplification are given. A set of fuzzy rules used by the GP to help accelerate convergence of the GP and improve the quality of the result are provided. For both RMs experimental methods of validating the decision logic evolved by the RMs are referenced to support the effectiveness of the data mined results.

References

[1] James F. Smith, III, "Fuzzy logic resource manager: decision tree topology, combined admissible regions and the self-morphing property", in *Signal Processing, Sensor Fusion, and Target Recognition XII*, Vol. 5096, I. Kadar, Ed. Orlando, SPIE Proceedings, 2003, pp. 104-114.

[2] S. Blackman, R. Popoli,: Design and Analysis of Modern Tracking Systems. Boston, Artech House, 1999, Chapter 11

[3] L.H. Tsoukalas, R.E. Uhrig,: Fuzzy and Neural Approaches in Engineering. New York, John Wiley and Sons, 1997, Chapter 5.

[4] H. J. Zimmerman: Fuzzy Set Theory and its Applications. Boston, Kluwer Academic Publishers Group, 1991, Chapter 1

[5] J.F. Smith, III, R. Rhyne, II,: "A Resource Manager for Distributed Resources: Fuzzy Decision Trees and Genetic Optimization." in Proceeding of the International Conference on Artificial Intelligence, IC-AI'99, Vol. II. H. Arabnia, Ed. Las Vegas, CSREA Press, 1999, pp. 669-675.

[6] D.E. Goldberg,: Genetic Algorithms in Search, Optimization and Machine Learning. Reading, Addison-Wesley, 1989.

[7] James F. Smith, III, ThanhVu H. Nguyen "Fuzzy Logic Based Resource Manager for a Team of UAVs," *North American Fuzzy Information Processing Society's Conference*, in press, IEEE Proceedings, Montreal, 2006.

[8] J. H. Holland,: Hidden Order How Adaptation Builds Complexity. Reading, Perseus Books, 1995, pp. 1-15.

[9] J.P. Bigus,: Data Mining with Neural Nets. New York, McGraw-Hill, 1996 Chapter 1

[10] J.R. Koza, , F.H. Bennett III, D. Andre, M.A. Keane: Genetic Programming III: Darwinian Invention and Problem Solving. San Francisco, Morgan Kaufmann Publishers, 1999, Chapter 2

[11] J. Gribbin, : Companion to the Cosmos. London, The Orion Publishing Group Ltd, 1996, p. 299.

[12] S. Luke, L. Panait: "Fighting Bloat with Nonparametric Parsimony Pressure." in Parallel Problem Solving from Nature - PPSN VII, 7th International Conference. Proceedings. LNCS Vol.2439. J.J.M Guervos, Ed.: Berlin, Springer-Verlag 2002, pp. 411-421

[13] J. F. Smith, III, R. D. Rhyne II,: "Fuzzy logic resource manager: tree structure and optimization" in: Signal Processing, Sensor Fusion, and Target Recognition X, Vol. 4380. I. Kadar, Ed., Orlando, SPIE Proceedings, 2001, pp. 312-323

[14] James F. Smith, III, ThanhVu H. Nguyen "Distributed autonomous systems: resource management, planning, and control algorithms," in *Signal Processing, Sensor Fusion, and Target Recognition XIV*, Vol. 5809, I. Kadar Ed., Orlando, SPIE Proceedings, 2005, pp. 65-76.