Adaptive Expert Systems for Indirect Coverage Control

Greg von Pless and Zack Butler
Dept. of Computer Science
Rochester Inst. of Technology
herrvp@gmail.com, zjb@cs.rit.edu

Abstract— **Herds of livestock, when left to their own devices, will forage for food in predictable patterns, and will often overgraze preferred areas while leaving other areas untouched. The field of grazing management looks to improve the efficiency of land use by moving the animals through different pastures at regular intervals, akin to coverage algorithms used in robotics but with non-uniform coverage and additional constraints due to the animals' natural behaviors. The knowledge of the field of grazing management is largely in the hands of ranchers and other domain experts, and as such has not been the subject of much computational study. In this work, we have created novel learning techniques for expert systems that use both off-line and on-line learning to generate efficient performance. The rules of the expert systems are initially developed using an evolutionary algorithm, and after deployment, an adaptive algorithm tracks the state of the system and updates rule weights to improve both coverage efficiency and animal stress levels. We present a series of results based on increasingly complex versions of simulated herd models that show improvements over unconstrained motion in each case, and suggest how the algorithm can apply to robotic coverage problems.**

I. Introduction

Rule-based expert systems are used in many domains to aid decision making based on rules derived from human knowledge about the domain. By using rules of the form "if [a given state is observed], then [do something]", the basic decision process of a human is mimicked and as such, these systems can be easily constructed by a domain expert. To use a system of this type, the current state of the environment is assessed with respect to the rules to arrive at a decision. In systems with an ongoing interaction with the physical world, the rules must take into account the relevant conditions, and must be general enough to handle the full range of states that may be encountered. To develop such systems, instead of relying solely on domain experts, it may be possible to use machine learning techniques to improve the rule base. This requires some way of assessing the quality of the decisions made by the system, so that the learning algorithm can judge whether and how to change the rule base.

In this work, we have developed new learning techniques for expert systems, applied to a coverage problem in the domain of grazing management. Traditionally, a herd of livestock is left to its own devices within a large fenced paddock, and will graze preferred areas, a simple technique known as continuous grazing. To improve the utilization of available land, the technique of intensive grazing involves moving the animals through a larger number of small paddocks, much like grid-based coverage algorithms used in robotics. Intensive grazing, however, requires a large amount of labor and fencing infrastructure compared to traditional methods. Recent results with virtual fences [4] suggest that computational techniques (hardware and software) can partially replace both infrastructure and labor. Virtual fences consist of collar devices placed on each animal that use GPS and computation to deliver stimuli when the animal is leaving the desired paddock area. On-board computation allows the fences to be automatically moved over time in a way useful for intensive grazing.

To determine when and how to move the animals through their environment, we have created a novel expert system framework under which rule sets are learned from experience (currently via simulations), both off-line and on-line. In the off-line portion, an evolutionary algorithm is used to develop useful rules for the expert system. Once the expert system is brought on-line, the weights of the rules are adjusted to enforce those which lead to better results. In this way, the system can adapt to the specific nature of a group in the particular environment.

A. *Background / Related Work*

A large body of prior work exists in the fields of both evolutionary expert systems and adaptive expert systems. Many methods of evolving expert systems have been developed, operating with various levels of complexity. We will briefly outline the nature of the expert system we will use and describe some relevant past research.

In a rule-based expert system, a set of rules is used, along with the current environment state, to perform inferencing and generate an output or a decision. Generally, each rule is of the form: if x, then y; in a rule of this form, x is referred to as the *antecedent*, and y as the *consequent*. Fuzzy expert systems make use of fuzzy sets in order to express rules at a higher level. Rules in a fuzzy expert system are generally of the form: if x is in A, then y is in B, where A and B are fuzzy sets with pre-defined membership functions. The degree to which y is in B is determined, through a process known as defuzzification, using the degree to which x is in A. In addition to generating the set of rules, the developer must also specify the fuzzy membership functions. The advantage of this approach over simple expert systems is that the rules read more like a person's reasoning. For instance, instead of a rule such as "If it is less than 40°F, then I should cover 90% of my body" the fuzzy system can use a rule such as "If it is cold, then I should dress warmly."

Several groups have looked into optimization of expert systems in different ways, either adjustment of parameters or development of completely new rules. For example, Perneel et al. propose a manner for performing parameter optimization on a tiered system [7], in which optimization is performed over the overall weighting of the tiers (priorities) of the rules, the weighting of the rules within the tiers, and the parameters defining the fuzzy membership functions. From the other direction, the goal of Drabarek et al. [5] was to evolve new rules from scratch for an expert system capable of performing diagnostics on a radar transmitter. Their chromosome consists of the rule antecedent and the rule consequent. The antecedent is a vector of integers representing a possible state of components in the system, and the consequent is a vector of integers representing which component should be checked. The whole chromosome is the concatenation of the antecedent part and the consequent part. An important component of both of these genetic algorithms is the calculation of fitness: how does one determine if a given rule weighting is good? In [7], a learning database is used that consists of a set of decision problems paired with desired decisions. In [5], a set of heuristics is used to determine if a given rule is acceptable. In our solution, we do not assume any "correct" decisions are known. Instead, the rule base is executed in simulation, and fitness is determined by metrics related to the final outcome rather than the decisions made by the expert system or the rules themselves.

Automated learning of general rules and fuzzy membership functions together is harder due to the size of the combined search space, but several approaches have been investigated. Amaral et al. use a serial approach: first, rules are evolved; then, memberships are evolved [2]. Separate genetic operators and fitness functions are required for each stage of the process in this case. In contrast, Akbarzadeh-T et al. propose a co-evolutionary approach to evolving fuzzy systems, wherein the rules are evolved via genetic programming, and the memberships are evolved by a genetic algorithm [1]. Here, search space complexity is lessened through the use of a binding matrix, which determines which rule sets combine with which membership function sets (not all combinations are tried). This method was largely successful on their test domain, and outperformed both a static expert system and an expert system consisting of uniform membership functions and rules evolved using genetic programming. Our solution is similar in intent, but uses a set of genetic operators that can modify the rule base and/or membership functions at the same time, and without requiring genetic repair.

Finally, some work has also taken place in the area of adaptive expert systems, in which rules are updated online to take into account new knowledge gained during execution. For example, adaptive-network-based fuzzy inference systems (ANFIS) [6] use artificial neural networks to learn both rules and membership functions based on desired decisions. As with most neural networks, it can be used offline or online to update as more information is obtained. However it does require "correct" decisions to use as training data, which are not available as such in our system.

II. Algorithm Design

The automated grazing management solution is comprised of three parts: the coverage algorithm, which uses the expert system to perform grazing management, the evolutionary algorithm, which is used to learn a good expert system given a fitness metric, and the adaptation algorithm that updates the rules during operation. The evolutionary algorithm and on-line expert system adaptation algorithm complement one another by performing global and local learning and adjustment, respectively. The evolutionary algorithm operates on a metric supplied at the end of testing, representing the overall performance of the system; the on-line adaptation algorithm operates throughout testing, making adjustments to the rules based on observed cause-effect relationships while it runs.

A. Coverage Overview

The technique used to move the herd throughout their environment is inspired by the indirect coverage algorithm described by Pirzadeh and Snyder [8]. In indirect coverage, the environment is split into a grid. The number of visits to each cell is stored; when deciding which cell to visit from the current one, the neighboring cell with the fewest total visits is chosen. Eventually, the whole environment will be visited at least once. Our algorithm likewise splits the field into a grid of large cells (each cell should be large enough to allow the herd to graze for a while before they must be forced to move), and keeps the animals within one cell using four virtual fences deployed along the edges of the current cell, each having infinite length. At a pre-defined interval, an expert system is used to calculate a score for the current cell and each cell within the current cell's 4-neighborhood. If the current cell's score is highest, then nothing is done; otherwise, the herd is moved to the cell with the highest score by slowly moving the locations of the virtual fences. This process is repeated indefinitely to move the herd throughout the environment.

The expert system itself consists of three parts: the rule set, the fuzzy membership functions, and an adaptation threshold value θ. The rules are used to calculate a score for a cell, as detailed below. Fuzzy membership functions are used to determine the extent to which the cell's raw data belong to the variables described in Table II.

The expert system's rules are all of the form "if *expression* then y," where y is a scalar amount to add (or subtract) from the cell's total score when *expression* evaluates to $true$ for that cell. The *expression* is a fuzzy expression composed of operators and operands. The simplest type of expression is a fuzzy membership query, such as "If VEGETATION is LOW", but these simple expressions can be combined to make more complicated expressions. The rules are naturally represented as trees, with operands at the leaves and operators at the root and other nodes. Table I describes the possible operators and Table II describes the fuzzy variables used in our system.

Three trapezoidal membership functions are stored for each variable, defining that variable's fuzzy membership in the sets LOW, MED, and HI. The only limitations imposed upon the definition of these functions are that the LOW

Operator	Syntax	Evaluation
and	*expr1* and *expr2*	$\min(\mathit{expr1}, \mathit{expr2})$
or	*expr1* or *expr2*	$\max(\mathit{expr1}, \mathit{expr2})$
not	not *expr*	$1 - \mathit{expr}$
current-cell	current-cell	1 if the cell being processed is the current cell, 0 otherwise.
reachable	reachable	1 if the cell being processed is directly reachable from the current cell, 0 otherwise.

TABLE I
OPERATORS FOUND IN EXPERT SYSTEM RULES.

Variable	Description
VEGETATION	The mean vegetation value for the cell.
WATER	The percentage of the cell consisting of water.
OBSTACLES	The percentage of the cell that is impassable.
EVENNESS	The difference in elevation between the highest and lowest points in the cell, expressed as a percentage of the maximum possible difference.
HAPPINESS	Perceived happiness level of the cattle in the cell based on the number of shocks recorded in the cell for the given time period.
CELL-AGE	How long this cell has been the current cell, or how long it has been since this cell was the current cell.
LINEARITY	How linear the path taken by the algorithm has been.
HERD-DIST	The ratio of the distance the herd has moved within a recent time interval to the maximum distance possible during the interval.

TABLE II
VARIABLES USED IN EXPERT SYSTEM RULES.

and HI sets may not overlap, and that for an input x, $\mu_{\text{LOW}}(x) + \mu_{\text{MED}}(x) + \mu_{\text{HI}}(x) = 1$. This is accomplished by using a vector of increasing x values for the corners of the trapezoids. The membership thresholds are set during the evolutionary process. An example is shown in Fig. 1.

B. Expert System Evolution

To create a useful set of rules for the expert system, the overall approach is a fairly standard genetic algorithm. In our framework, an entire rule set is the gene to be evolved, and we use specialized crossover and mutation operators to ensure that all resultant genes represent valid systems. Fitness of a rule set is calculated through simulation as described below. We start with a small set of hand-made expert systems and run these through a number of generations until we detect convergence of fitness; within each generation individuals are selected, reproduced with crossover, and potentially mutated.

Specifically, a gene contains all the components of the expert system as listed above. Each rule is composed of two parts: the antecedent and the consequent. The antecedent is an expression containing operators and operands; since an operand may be an expression in and of itself, the antecedent is represented by a tree, as is generally done in genetic programming. The consequent is a scalar value, and so is stored as such. The genotype maintains a set of pairings of antecedent trees and consequent values to represent the rule set. The membership functions are a vector of floating-point values, and the threshold θ a single floating-point value.

Crossover is performed separately on rule sets and membership functions; when two genes undergo crossover, both their rules and membership functions are affected. For the rule sets, 1-point crossover is used at the set level (i.e., the first n rules are kept, and the remainder are swapped with the other parent, noting that different rule sets may be of different size). Swapping of parts of rules occurs during mutation. 1-point crossover is also used for membership functions; however, the only allowable cut-points are those occurring between individual functions. This limitation prevents the need for post-crossover genetic repair, because, within a specific function definition, the function parameters must be non-decreasing. Finally, the adaptive θ threshold undergoes 1-point crossover at the bit level.

For mutation, we implemented several domain-specific operations, each of which changes a single gene (rule set):

- New-Rule: create a new rule by randomly generating a rule tree from the operators and operands available.
- Delete-Rule: delete a randomly-chosen rule.
- Replace-Rule: replace a randomly-chosen rule with a new randomly generated rule.
- Swap-Subtrees: swap subtrees of two different rules.
- Mutate-Consequent: set the consequent of a randomly chosen rule to a new random value between -4 and 4.
- Mutate-Theta: alter one bit in θ.
- Replace-Membership: replace a membership function vector with a new one.

At each generation, for each member of the population, one of these functions is executed with low probability.

To determine the fitness of a particular gene, we simulate using the rules to control a herd of animals. The simulator was developed for previous work [4] and uses a potential-field-based method to determine animal behavior. Animals are generally attracted to the rest of the herd while repulsive forces prevent them from clustering too tightly. Contact with a virtual fence results in sharp turns and stress, which in turn causes higher speed motion for a short time, as evidenced by physical experiments [3]. Additional behavioral factors such as differing levels of vegetation, thirst and elevation (animals prefer to travel at constant elevation rather than up and down) are also included. Our experiments included runs with and without these factors as described below. In all cases, we run a simulation of 150 hours of grazing time, after which the resultant coverage and cumulative animal stress determines fitness of the set. Coverage was measured by comparing the distribution of vegetation levels to a preferred distribution centered around a medium level of growth (specifically,

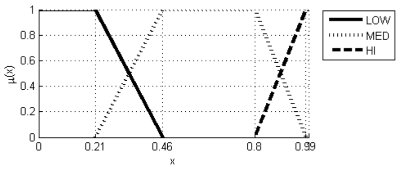

Fig. 1. Fuzzy membership functions corresponding to the array $M = [0, 0.21, 0.46, 0.8, 0.99, 1]$ — these values were created as the memberships for EVENNESS in Experiment 2.

we use the χ^2 value of the actual distribution relative to the preferred). This preferred distribution was used to favor patterns that exhibit neither overgrazing nor undergrazing. Examples are shown in the experimental results. Animal stress was measured by the number of stimulus events in a given time period u, but for fitness calculations, the value e^u was used to limit the amount of stimulus used. These can be used independently (for example, for robots, stress would not be an issue) or multiplied together (since for both metrics, small values are desirable).

C. On-line adaptation

The adaptive algorithm attempts to determine which rules are most responsible for success (and inhibitory to success) and reinforces (or diminishes) their effects. It does this by maintaining a directed graph associating environment states with perceived happiness levels of the herd; thus, this graph provides a rudimentary representation of cause and effect. The current cell state at time t is represented by the vector

$$S_C^t = [V, O, W, A, E]$$

where V, O, W, A, and E are integers denoting the fuzzy set (LOW = 0, MED = 1, HI = 2) to which the variables VEGETATION, OBSTACLES, WATER, CELL-AGE, and EVENNESS, respectively, *most* belong. Likewise, S_H^t is the set to which HAPPINESS most belongs. The following multi-sets hold the cumulative state:

$$S_C^* = \bigcup_t S_C^t, S_H^* = \bigcup_t S_H^t$$

The adaptive graph $G_{S_C^* \rightarrow S_H^*}$ is a weighted, directed graph mapping cell states to perceived happiness with weights indicating the number of occurrences of a specific mapping. At a set time interval (once per five simulated minutes in our experiments), the current state (S_C^t, S_H^t) is computed and the corresponding edge of G is given increased weight (or created, if it did not previously exist). If the weight of the edge is greater than θ, this is a significant relationship, and any relevant rules should be updated. (Recall that θ is set via the evolutionary process rather than a priori.)

The relevance of a rule to a cause-effect relationship is measured by determining the similarity between the cause (environment state S_C^t) and the state the rule attempts to match. For each rule, each operand is compared to S_C^t to compute a similarity d as follows: if the operand matches the relevant value in S_C^t, d is incremented, except that if the operand appears as the argument of a *not* clause, then d is increased if the state does not match the value in the operand. This is repeated for all operands in the rule, computing a similarity value only — that is, the presence of non-matching clauses does not detract from the similarity score. Then, the consequent value of the rule is either increased if the happiness level S_H^t is HI or decreased if S_H^t is LOW, with the magnitude of the increase/decrease based on d (so that rules that do not match the current state are unchanged). We use an exponential function of d to modify the consequent value, so that the change is more drastic when its current value is significantly different from the ideal. This behavior is intended to mimic a human modifying his solution to a problem: in the case that the solution is "way off," he may make large changes; however, if the solution is close, but not quite right, he will most likely only tweak it slightly.

III. EXPERIMENTAL RESULTS

To determine the utility of our approach, we conducted a number of experiments exploring various aspects of the system. Specifically, we trained and tested the expert system using a herd simulator, starting with very simple herd behavior in a simple environment without using on-line rule adaptation. We then increased the complexity of the animal behavior and finally added adaptation over a series of tests. Due to space considerations, here we present three experiments from the full series of tests that were run. More details and the full set of experiments can be found in [9].

Experiment 1: Simplified Behavior Model

This experiment uses a large subset of the simulated animal behaviors as a proof-of-concept of the evolutionary expert system. In this set of simulations, in addition to wandering and herding behaviors, we simulate different vegetation types, with varying nutritional values and regrowth rates, and elevation is made a factor. Finally, we use attractor locations (areas of the environment that the animals prefer to be in) on the same side of the field, in order to make the cattle less likely to cover the field naturally. The major factor that was left out is the animals' need for water on a regular basis. Using this configuration of the simulator, uninhibited grazing yielded an average coverage-based fitness of $\chi^2 = 5.453$, and a best fitness of $\chi^2 = 3.693$ over five trials (see Figure 2). Two evolution experiments were then performed: one using the coverage-based fitness function, and one using the stress-based fitness function.

After 15 generations, the evolutionary algorithm with coverage-based fitness yielded an expert system with a fitness of $\chi^2 = 0.110$. This value persisted for three more generations, so the algorithm was considered to have converged. Figure 3 shows the final vegetation levels and histogram for the fittest expert system. The evolved system contained five rules with a primary focus on visiting cells with more (or higher quality) vegetation. The other rules were regarding

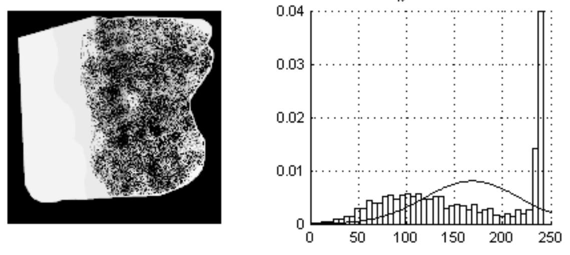

Fig. 2. Experiment 1 control results — best χ^2 fitness without management. Left: vegetation levels after 150 hours (higher intensity corresponds to more vegetation); right: final vegetation distribution histogram and ideal distribution curve.

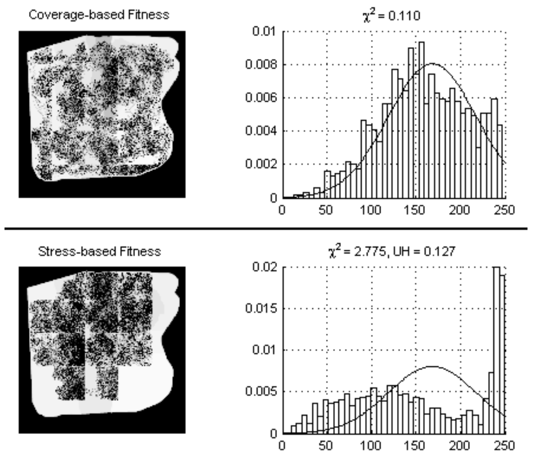

Fig. 3. Experiment 1 results. Top: best χ^2 coverage-based fitness; bottom: best stress-based fitness.

cattle happiness, and initially seemed to be aimed at making the cattle miserable; however, it was determined during this experiment that the cow happiness estimate calculation was virtually always yielding HI happiness values. Thus, one happiness rule was almost always in effect, causing a blanket reduction in all scores (the happiness calculation was corrected before switching to the stress-based fitness metric).

Using the stress-based fitness metric, the evolutionary algorithm converged in 5 generations to a fitness of 0.127 (i.e., an average of 1 stimulus every 7–8 seconds). The coverage metric for the final expert system was $\chi^2 = 2.775$ (see Figure 3). The rule set consisted of 8 rules that assign nearly equal weight to HAPPINESS and VEGETATION in addition to rules that favor HI values of CELL-AGE. We note that low stress levels may be difficult for the system to achieve since it is required to contain the animals within a single cell at all times.

Experiment 2: Full Simulator Functionality

This experiment pitted the grazing management algorithm against the full complexity level offered by the simulation; specifically, water sources were added to the field along with clumps of trees, and a new, more realistic elevation map was developed. Figure 4 depicts the field layout and elevation.

The management task is significantly more difficult with the addition of water sources and the related simulated cattle behavior. A majority of the management algorithm's cells do not contain a water source, so the system must learn to rotate the cattle among these cells, and then return them to a cell containing water when the cattle become thirsty.

Beginning with this experiment, the evolutionary algorithm was prevented from generating rules whose antecedents consisted only of the *current-cell* operator. Allowing rules of this form to be generated significantly reduces the likelihood of producing complex rules. This experiment included three separate runs of the evolutionary algorithm: as well as the coverage-based and stress-based fitness functions, we also performed evolution using the combined fitness metric.

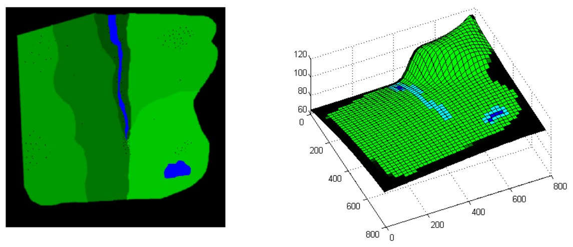

Fig. 4. Left: Aerial view of the final simulation field incarnation; right: 3-D view illustrating elevation changes throughout the field. Water is present at the lower right (the dark blob) and vertically through the top center.

Results: Continuous grazing in this field with complete animal behavior yielded an average fitness of $\chi^2 = 6.887$ over five trials, and a best fitness of $\chi^2 = 5.849$ (see Fig. 5). Because the combined fitness is calculated as $f = \chi^2 e^u$, and $u = 0$ when the cattle are allowed to roam free, these values are also the mean and best combined fitness values.

Coverage-based fitness converged on a fitness of $\chi^2 = 3.855$ after 8 generations. Figure 6 shows the final vegetation levels and distribution histogram for the winning expert system. The fittest system's rule set contained nine rules, but interestingly, none of them used the VEGETATION variable. Rather, through concern for cattle restlessness (measured through HERD-DIST) and a reduction in the current cell bonus, the expert system keeps the cattle moving between cells, thereby avoiding overgrazing. A large negative score modification resulting from "not reachable" prevents the system from pushing the cattle through the river.

After 9 generations, stress-based fitness converged to a fitness of 0.013 (an average of approximately 1 stimulus per 77 seconds), with a very poor coverage metric of $\chi^2 = 24.836$ (see Figure 6). The rule set created was very similar to the one generated via coverage-based fitness, but with two significant differences. First, rather than a large penalty for MED HAPPINESS, there is a large bonus; second, the blanket reduction to the current cell bonus is so large that the current cell suffers a large penalty. So, the system rarely remains in the same cell; however, because of the significant bonuses to MED WATER and MED HAPPINESS, the system will cycle back and forth between the same few cells containing moderate amounts of water. This is illustrated well in the image of the final vegetation levels: vast portions of the field are untouched, while areas around the pond and river are

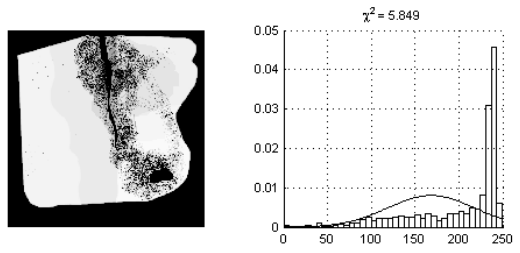

Fig. 5. Experiment 2-3 control results — best χ^2/combined fitness without management.

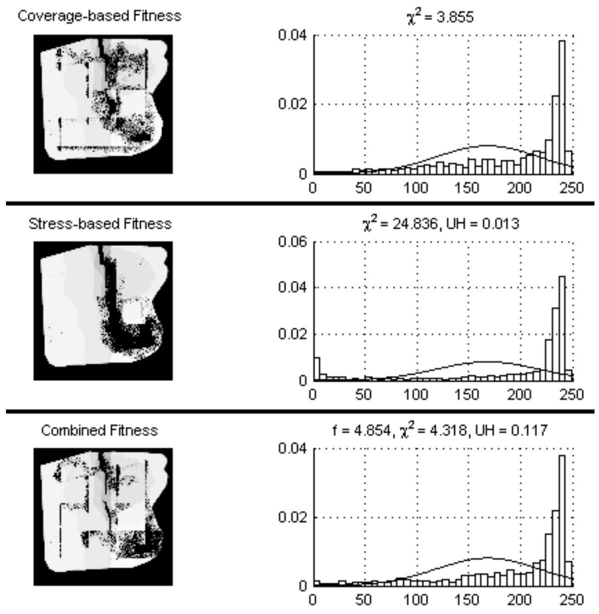

Fig. 6. Experiment 2 results. Top: best χ^2 fitness with management; middle: best stress-based fitness; bottom: best combined fitness.

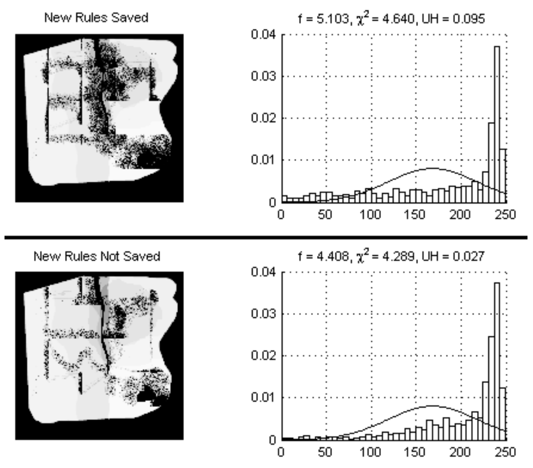

Fig. 7. Experiment 3 results. Top: best fitness with adapted rules saved; bottom: best fitness with adapted rules discarded.

extremely overgrazed.

Finally, the combined fitness function converged in 15 generations, yielding an expert system with a combined fitness of 4.854. The coverage metric for this system was $\chi^2 = 4.318$; the stress-based metric was 0.117, or an average of approximately 1 stimulus every 8–9 seconds (see Figure 6). The evolved rule set is as follows:

1 (HAPPINESS is HI) $\rightarrow +2.0$
2 (WATER is MED) $\rightarrow +3.0$
3 (VEGETATION is HI) $\rightarrow -0.75$
4 (CELL-AGE is LOW) $\rightarrow -1.0$
5 (HAPPINESS is LOW) $\rightarrow -3.0$
6 (current-cell and HERD-DIST is LOW) $\rightarrow -3.14$
7 (not reachable) $\rightarrow -10.0$

This rule set is essentially a combination of the previous two rule sets, but without some superfluous rules — for each of WATER, VEGETATION, and CELL-AGE, only one rule is present where two or three were used in the other systems. This system suffers two flaws: first, that it assigns a slight penalty for HI VEGETATION; second, that it waits too long to respond to certain behavior — LOW HAPPINESS and LOW HERD-DIST — that can be indicative of increased stress. Reacting to LOW HERD-DIST is difficult as this may indicate that the cattle are grazing happily, or it could indicate that they are clustered against a fence, trying to escape. This system takes it to mean the latter; however, at this point the cattle should probably have already been moved.

The results of this experiment demonstrate the necessity of the combined fitness function; stress-based fitness, especially, is not effective on its own, yielding an atrocious χ^2 score. Furthermore, combined fitness is shown here to successfully improve coverage over continuous grazing without causing high stress levels among the cattle.

Experiment 3: On-line Adaptation of the Expert System

This set of experiments features the introduction of the on-line adaptation portion of the grazing management algorithm. The animal simulation reached its full complexity previously; no new functionality was added during these tests. Fitness was calculated using the combined fitness metric. Two methods for handling the adapted rules with respect to the evolutionary algorithm were explored. In the first case, at the completion of a simulation, the adapted versions of the rule sets were stored, so that the evolutionary algorithm then operated on these new rules; here the intention is to pass what is learned on to later generations. In the second case, the new rules were not saved; rather than passing along what is learned, the intent here is to evolve systems most capable of adapting to the situation at hand.

Results: The previous section's baseline continuous grazing results are also used for comparison here (see Figure 5). Recall that the baseline best combined fitness was 5.849. In the case wherein adapted rules were saved before performing crossover and mutation, convergence took 14 generations, yielding a best fitness of 5.103 (see Figure 7); coverage-based fitness was $\chi^2 = 4.640$, and stress-based fitness was 0.095. The winning rule set follows.

1 (WATER is HI) $\rightarrow +0.5$
2 (WATER is MED) $\rightarrow +3.0$
3 (HAPPINESS is HI) $\rightarrow -0.99$
4 (CELL-AGE is HI) $\rightarrow +1.31$
5 (HAPPINESS is LOW) $\rightarrow -3.0$
6 (HERD-DIST is HI) $\rightarrow +0.46$

The system's θ was 53, meaning that in order to respond to an observed cause-effect relationship, that relationship would have to be seen 54 times; with the system running once every 5 minutes, that equates to an absolute minimum of 4.5 hours before rule updates, and likely even more rare.

Without saving adapted rules, a best fitness of 4.408 was obtained in 7 generations. This system yielded a coverage-based fitness of $\chi^2 = 4.289$, and a stress-based fitness of 0.027. Figure 7 depicts the winning system's final field state and vegetation distribution. The resulting rules:

1 (WATER is HI) $\rightarrow +0.5$
2 (WATER is MED) $\rightarrow +3.0$
3 (HAPPINESS is HI) $\rightarrow +1.0$
4 (HAPPINESS is MED) $\rightarrow -0.75$
5 (HAPPINESS is LOW) $\rightarrow -3.0$
6 (current-cell and HERD-DIST is LOW) $\rightarrow +4.0$
7 (not current-cell and HERD-DIST is HI) $\rightarrow -2.5$
8 (not reachable) $\rightarrow +0.79$

Rule 8 is very curious: without a large penalty for unreachability, the system may try to push the cattle through the river. In the simulator, water is considered to be an obstacle, so the cattle receive large repulsive forces if they attempt to enter it. With a virtual fence pushing them the toward the water, the cattle may be forced across the river, however, the entire herd is shocked throughout the process. Visual inspection of the final field state indicates that the cattle were indeed forced across the river at least once, and possibly twice, but the overall stress levels were still low, so if a penalty had been in place for unreachability, this system may have been extremely successful.

In stark contrast to the previous system's $\theta = 53$, this system's θ was 4, allowing learning to occur as often as once every 20 minutes simulation time. The behavior of this system, therefore, most likely changes a great deal during simulation. Evidence of this can once again be seen in the final field state. There are definite regions of overgrazing on the right side of the field near the herd's starting location. On the left side of the river, overgrazing occurs much less frequently, indicating that by the time the cattle had been moved to that side, on-line adaptation had taught the system to better avoid this undesirable behavior. An explanation for the disparity between the θ values is that saving the adapted rules carries with it an inherent risk: if the adapted rules are good, then saving them is useful, as they will become available in the genetic pool; however, if the adapted rules are poor, then the genetic pool becomes poisoned by these rules. That the system evolved through saving adapted rules had a θ value of 53, in effect forcing minimal on-line learning, is evidence of this suspected behavior; the evolutionary algorithm produces systems that rarely adapt in an attempt to prevent the poisoning of the genetic pool.

IV. DISCUSSION AND FUTURE WORK

This first foray into the field of automated grazing management using dynamic virtual fences has proven to be a successful one. The proposed evolved and adapting expert system was shown to consistently improve coverage over traditional continuous grazing. Furthermore, both the procedure for evolving expert systems and the on-line adaptation algorithm that were designed and developed for this work proved successful: the evolved expert systems yielded superior results to hand-made rule sets and the addition of the adaptation algorithm further improved these results.

There are a number of ways in which this work could be expanded and improved. First of all, while we have built systems for simulated animal behaviors, it may also be applied to systems such as robotic coverage with very different underlying fitness requirements, and it would be interesting to see performance on such different tasks. Alternately, the use of a more complex and accurate cow model would greatly increase the direct applicability of the developed management systems for deployment on real cattle. Our current simulations, as they simulate animal behavior on a second-by-second basis, take a long time to execute, which in turn slows the evolutionary algorithm. Although this may be acceptable in situations where the rule set is developed once and used (and adapted) over time, it does slow the pace of development. With a more efficient simulation, we may be able to simulate a variety of environments to calculate fitness and produce a more flexible rule set.

Finally, we believe that this type of technique may be applicable to other coverage and related problems in robotics. For situations such as surveillance or environmental monitoring, we wish to cover the entire environment, but the importance of visiting certain areas may be non-uniform (e.g. some places are more exposed and therefore need more frequent surveillance), and may further vary over time (for example, as events of interest are noticed in different places). The adaptive expert system as developed here can compute effective initial coverage patterns at a high level, leaving local path generation to a traditional planner, and continue to update these patterns as the situation evolves.

Acknowledgements

This work was supported in part by NSF grant IIS-0513628.

REFERENCES

[1] M.-R. Akbarzadeh-T, I. Mosavat, and S. Abbasi. Friendship modeling for cooperative co-evolutionary fuzzy systems: A hybrid ga-gp algorithm. In *22nd Int'l. Conf. of the North American Fuzzy Information Processing Society, 2003.*, pages 61–66, 2003.

[2] J. F. M. Amaral, R. Tanscheit, M. A. C. Pacheco, and M. M. R. Vellasco. Evolutionary fuzzy system design and implementation. In *Neural Information Processing, 2002. ICONIP '02. Proceedings of the 9th International Conference on*, volume 4, pages 1872–1876, 2002.

[3] D. M. Anderson, B. Nolen, E. Fredrickson, K. Havstad, C. Hale, and P. Nayak. Representing spatially explicit directional virtual fencing (DVF™) data. In *24th Annual ESRI Int'l User Conference*, 2004.

[4] Z. Butler, P. Corke, R. Peterson, and D. Rus. Dynamic virtual fences for controlling cows. In *Int'l Symp. on Experimental Robotics*, 2004.

[5] J. Drabarek, R. Wirski, and W. Madej. Genetic algorithms applied to hybrid expert systems. In *Electronics, Circuits and Systems, 2002. 9th International Conference on*, volume 3, pages 1263–1266, 2002.

[6] Jyh-Shing Roger Jang. ANFIS: Adaptive-network-based fuzzy inference system. *Systems, Man and Cybernetics, IEEE Transactions on*, 23(3):665–685, 1993.

[7] C. Perneel, J.-M. Themlin, J.-M. Renders, and M. Acheroy. Optimization of fuzzy expert systems using genetic algorithms and neural networks. *Fuzzy Systems, IEEE Transactions on*, 3(3):300–312, 1995.

[8] Amir Pirzadeh and Wesley Snyder. A unified solution to coverage and search in explored and unexplored terrains using indirect control. In *Proceedings of IEEE ICRA*, volume 3, pages 2113–2119, 1990.

[9] G. von Pless. Automated grazing management. Master's thesis, RIT, 2008.