SmartSwarms: Distributed UAVs that Think

Dr. Armand Prieditis
Dr. Mukesh Dalal
Andrew Arcilla
Brett Groel
Michael Van Der Bock
Richard Kong
Lookahead Decisions Incorporated
prieditis@lookaheaddecisions.com

Abstract

Unmanned Aerial Vehicles (UAVs) have demonstrated tremendous capability in recent military operations. Recently swarm technology has been suggested as a possible solution to automatically control and coordinate multiple UAVs. The idea behind a swarm is that simple local rules that govern the behavior of individual entities can lead to complex *emergent behavior* of the system as a whole. Although such systems have achieved limited success in simulated applications, finding good rules can be difficult for humans. Moreover, such rules can result in odd behavior or unnecessarily long missions. This paper describes a swarm-based multi-UAV system, called SmartSwarms, using a radically different approach: instead of operating with human-defined rules, each individual reasons using Simulated LookAhead (SLA), thus incorporating a model of its world and nearby entities in decision-making. Our results show that this approach can improve swarm behavior in UAVs. SLA is affordable, scalable to a large number of UAVs, deconflicts in real-time, learns over time, is interoperable, reusable, fault tolerant, and error tolerant, and can handle uncertainty.

Keywords: computer generated forces, agent-based combat modeling, real-time decision-making, swarms, UAVs, UGVs.

Introduction and Motivation

Unmanned Aerial Vehicles (UAVs), including Predator, Hunter, Shadow, and Pioneer, have demonstrated tremendous capability in recent military operations. They offer several advantages, typically categorized as "the dull, the dirty, and the dangerous." They can increase unit effectiveness and multiply force by surveying an area better than ten or more human sentries and have longer persistence ("the dull"); they can reconnoiter in areas contaminated by nuclear, chemical, or biological agents without risk to humans ("the dirty"); they can suppress enemy air defenses in high risk operations currently flown by manned EA-6's or F-16s with less support aircraft, thus eliminating the risk of loss of life of an air crew ("the dangerous"). Moreover, UAVs cost less to acquire and support, have a greater potential for survivability, enable new CONOPS, and can act in parallel as an extended sensor network, providing a more accurate picture of a larger battlespace. In short, UAVs are poised to offer significant change in the way the military conducts operations.

However, pilots who fly the UAVs are often overwhelmed with sensory information and have difficulty deconflicting a UAV while focusing on the mission at hand. At the same time, the pilot is not able to receive the right sensory cues—cues that are normally available to the pilot, but are lacking at the remove location.—to aid in deconfliction. This can result in mishaps and failure to complete the mission.

Worse still, current UAVs require at least one operator per UAV, despite technological advances that make it possible to deploy hundred (if not thousands) of inexpensive. This requirement not only increases expense, but makes coordination among UAVs more difficult.

Recently swarm technology has been suggested as a possible solution to automatically control and coordinate multiple UAVs. Swarms consist of a large number of distributed, parallel-acting individual entities coupled with primitive communication mechanisms such as chemical markers. The idea behind a swarm is that simple local rules that govern the behavior of individual entities can lead to complex *emergent behavior* of the system as a whole. For example, it might be possible to conduct a search and destroy operation. Rules include ideas such as "avoid areas already searched" or "avoid UAVs within a certain radius."

Although such systems have achieved limited success in simulated applications, finding good rules can be difficult for humans. Moreover, such rules can result in odd behavior or unnecessarily long missions.

This paper describes a swarm-based multi-UAV system, called SmartSwarms, using a radically different approach: instead of operating with human-defined rules, each individual reasons using Simulated LookAhead (SLA), thus incorporating a model of its world and nearby

entities in decision-making. In short, the individuals in our approach are capable of limited thinking rather than responding with reactive rules. Our results show that this approach can improve swarm behavior in UAVs. SLA is affordable, scalable to a large number of UAVs, deconflicts in real-time, learns over time, is interoperable, reusable, fault tolerant, and error tolerant, and can handle uncertainty.

Previous Approaches to Simulating Intelligent Behavior

The primary approach to intelligent behavior is *rule-based.* In the rule-based approach, an interpreter attempts to match the current situation to a particular rule, which then "fires." This "firing" can either result in a direct action in the simulation system or subgoals in the interpreter, which are then matched for other rules. In either case, the ultimate result is an action that it taken. Current rule-based approaches to modeling behavior include:

- **Production Rules.** Production rules are declarative rules with triggers and actions associated with the triggers. The actions may be local to the production rules—actions necessary to decide what to do in the application. TacAir-Soar is based on such rules. Using a fixed-size memory and a forward-chaining rule-based interpreter (Soar), it simulated fixed-wing and rotary-wing aircraft pilots flying combat and reconnaissance missions for STOW-E 1995 and STOW-97.
- **Finite-State Machines (FSM)**. A finite state machine consists of a list of states, commands that can be accepted at each state, actions associated with each command, and triggers for each action. The triggers are in the form of rules.
- **Agent-Based.** An autonomous agent capable of making decisions and communicating with other agents. The agents themselves are driven by rules. In fact, most autonomous agents are nothing more than a set of distributed production rules firing in parallel.
- **Logic-Based.** This approach uses propositional, predicate, or higher-order logic to represent rules. Sometimes theorem proving is used to produce a decision based on multiple rule "firings" much as with production rules. Most practical logic-based systems use the Horn Clause form of rules, the most complex yet tractable representation. As such, Horn Clauses have severe limits in expressiveness.

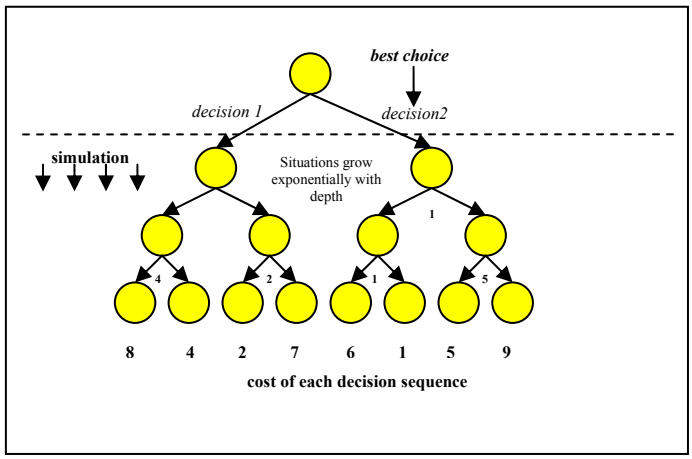

Figure 1 Lookahead Explicitly Builds a Decision Tree; The Best Choice is One that Minimizes Backed-Up Cost

- **Neural Networks.** This approach uses non-linear decision functions to choose actions. Inputs from the state are encoded as real-valued data; the output represents the action choice. The neural net is typically trained from data consisting of inputs and appropriate decision. The rule is implicitly encoded as a non-linear decision surface.
- **Genetic Algorithms.** This approach uses the process of evolution to choose the "fittest" set of inputs. The result is a rule that represents the best choice for a given set of inputs. In short, the Genetic Algorithm is not necessarily used to make decisions; it produces rules, which are then used to make decisions.

Our Approach to Intelligent Behavior

In contrast to the rule-based approach for decision-making, our approach to decision-making is *model-based*. This approach involves the forward application of a *model* of each decision—its conditions and effects—and events that can take place in the world. In other words, the model is the simulation model, which has already been developed for the application. Used by world-championship chess-playing programs, this approach can be summarized in the following three steps:

1. **Simulate** the effects of each decision forward in time by building a lookahead tree of possible decisions.
2. **Evaluate** the expected cost of each decision sequence in that tree.
3. **Choose** a decision that minimizes the expected cost.

For example, Figure 1 shows a lookahead tree where the object is to minimize the expected cost. The nodes of the

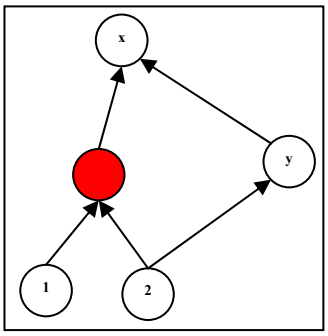

Figure 2 Rule-based vehciles (1 & 2) will attempt to move along the shortest path to their destination (x); model-based vehicles recognize that both will arrive at their destination faster if the second vehicle moves along a seeminlgy longer path (y) rather than take a shorter path through the choke point (shaded/red node)

tree represent situations and the arcs represent decisions or events that change the situation. Simulation predicts the expected outcomes below each of the two major decisions (at the top); the tree is generated explicitly through the application of decisions, which change the situation. The leaf nodes (nodes at the bottom of the tree) represent outcomes in terms of cost. According to decision-theory, the second decision is the best one to make. In short, model-based decision-making combines forecasting (predicting what the future might be like based on the model) with decision theory. For example, a

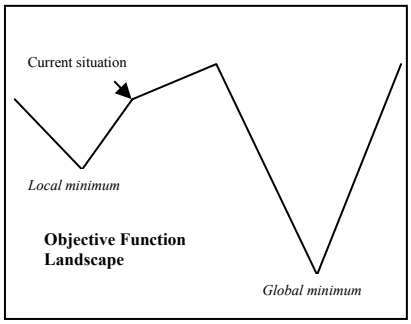

Figure 3 A two step lookahead cen see beyond the local maximum

weather forecast might call for rain; the decision to make is whether or not to take along an umbrella; decision theory says that the best decision is the one with the maximum expected utility (in this case not getting wet).

Advantages of Our Approach

Our approach has several advantages over the rule-based approach. Specifically, it is:

- **Telescopic.** It takes into account the *future* impact of each decision. In contrast, rules are *myopic* and often result in poor long-term decisions. For example, a three step lookahead in Figure would show that the final arrival time for both vehicles will be faster if the right vehicle takes a seemingly longer path to its destination (x). This is because the shaded node is a congestion point: according to the standard shortest-path dispatch rule, both vehicles will attempt to move into that node and one will end up waiting for the other to vacate that position. As such congestion frequently occurs with multiple vehicles, lookahead can automatically choose vehicle routes that avoid congestion because it can *anticipate* congestion. The more vehicles, the more likely the congestion and the more lookahead helps. Intuitively, lookahead is better able to evaluate the long-term impact of a decision. Mathematically, lookahead is able to overcome local minima in the objective function. For example, Figure shows that a two-step lookahead suggests a seemingly uphill move in anticipation of the global minimum, which lies beyond the peak. In contrast, a single-step lookahead will suggest a downhill move, which appears good, but will result in a local minimum.
- **Expressively Powerful.** It incorporates a model of uncertainty—decisions that have uncertain effects or random events. In contrast, rules have extreme difficulty expressing how to deal with such situations.
- **Real time.** It can make decisions within the time constraints of the underlying application. Complex rules, especially those requiring theorem proving or an interpreter, might not be real-time and hence have limited applicability.
- **Anytime.** It looks ahead as far as decision-making time permits: the more time available to make the decision, the better the decision. Greater decision-making time allows a deeper lookahead, which better gauges the long-term impact of each decision. For example, Figure 2 shows that chess ranking increases with chess lookahead for chess programs. In contrast, rules do not take advantage of any additional decision-making time that might be available.
- **Principled.** It operates from the underlying principles of the simulation model rather than ad hoc rules, which may not match the model. In general, it is difficult for a finite set of rules to capture a complex simulation system. We sidestep this problem by actually running the simulation system forward in time to make

decisions for intelligent entities. Once the simulation model is built, no additional work is required to apply model-based decision-making.

- **Easily deployable.** The model can be built rapidly at relatively low cost, without requiring great expertise. In contrast, rules require deep knowledge of the application. As a result, good rules are often difficult or expensive for an expert to articulate: many people could specify the domain, but few can specify the behavior require by intelligent entities. Moreover, a new rule is required for each situation, thus further increasing the cost of developing a rule base.
- **Flexible and easily maintainable.** When the simulation model changes, the decisions that result from it automatically change, with no additional effort. In contrast, rules might require change if the model changes. Maintaining a set of fixed rules is often a nightmare, one that results in spaghetti code with rules typically at odds with each other.
- **Optimal.** Theoretically, model-based decision-making produces a decision that is optimal relative to the underlying simulation model. This is because it simulates the model forward in time. Optimal decisions not only provide strong opponents (which are good for training), but they provide a trainee with the actual optimal choice for their own decisions—important for learning good decision-making skills.

For all of these reasons, model-based decision-making has dominated over the rule-based approach in many applications, ranging from chess to scheduling. We believe that it will similarly dominate the generation of realistic behaviors for intelligent entities in simulation.

Current Results

We are still generating results at press time, but we have developed and tested two different UAV environments. The first is a two-dimensional deconfliction environment consisting of 28 UAVs in a 6x6 grid. The task is to move each UAV from a given location to a target destination (selected at random) and then return to the start location. With 28 UAVs and only 36 locations, the complexity and congestion is considerable. Figure 3 shows SLA performs surprisingly well: with deep lookahead, it attains a 70% improvement over the standard dispatch rule (route along the shortest path to the destination), where the cost is the sum of path lengths over all UAVs. In other applications, different performance measures might be used. For example, in a military application, it might include a measure of mission success.

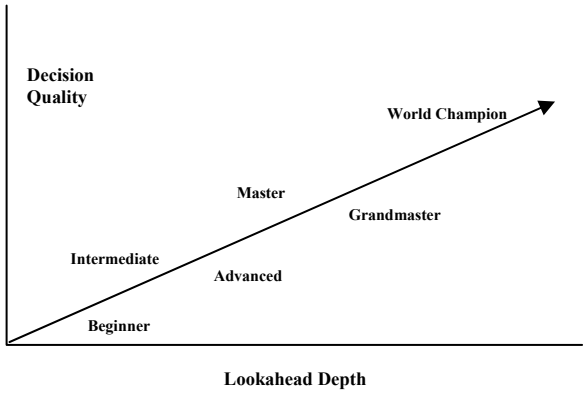

Figure 2 Decision Quality Improves with Lookahead Depth

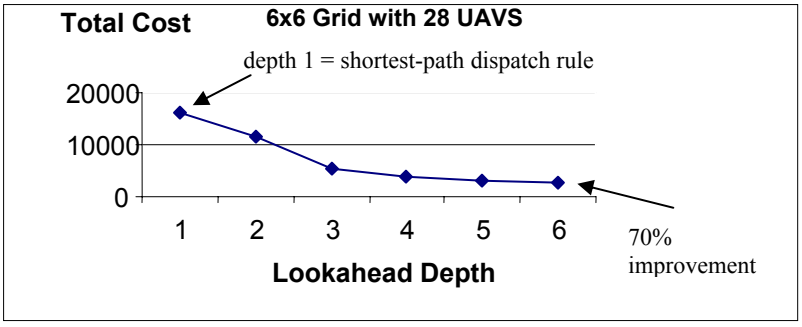

Figure 3 2D UAVs: The task here is to route UAVs as rapidly as possible while avoiding collisions; 1000 random destinations were used

The second is a 10 x 10 x 10 grid of discrete locations with 20 UAVs (each UAV occupies one discrete location). We compared SLA against a standard rule-based approach to controlling this swarm: follow the shortest path to destination whenever possible. If the shortest path is blocked, the next shortest path is found. We used 200 random situations. If problem can't be solved by either algorithm within 100 steps, we discarded it. For our algorithm, we used a lookahead depth of 10. In either case, we measured the average distance to colve the problem. Oasys resulted in an average of 19.9 moves and the shortest path rule resulted in 21.91 steps to goal. This translates to a 9.2% improvement for each UAV. With multiple UAVs this translates to significant savings in fuel. With a more dense space, the results will be even better.

Architecture

We have developed an interoperable architecture built on SLA. This architecture, called the Oasys system, consists of the following components:

- The **Supervisory** and **Advisory Controllers** and the **Exception Handler** cause actions to take place in the real world through **Action Adaptors** (our interface to the application-level **Execution Systems**).
- The **Execution Systems**, in turn, apply the actions in the simulated or real world.
- Application-level **Sensors** and other **Data Sources** sense that change in the world and communicate it to our **Sensor Adaptors**, which feed into the Data **Store**, the central clearinghouse for all of Oasys's information.
- The **Data Store** is used by all of the vertical components. For example, it is used by the **Visual Model Builder** to store the model, by the **Simulator** to run the model forward in time, and by the **Monitor** to display key information to the user.
- **Event Playback** allows the user to replay important events in the **Data Store**.
- **Report Writer** summarizes key performance indicators over many runs, actual or simulated.
- The **Model Learner** learns a model from historical data in the **Data Store**.
- The **Decision Learner** learns to make better decisions over time by analyzing past decisions.

The Oasys 3.0 architecture, shown in Figure 4 is distributable and extensible. There are several internal APIs and one external API. This external API is used by the execution system to communicate with Oasys. The API consists of several functions that allow the execution system to notify the Oasys of state changes and decision requests. State changes are reflected through monitors, which are essentially triggers that the execution system can fire when an important event takes place. Decision requests are made when a specific decision is needed. The Oasys API is implemented using Microsoft's .NET framework, but can easily be adapted to support other architectures such as COM and DLL. Oasys 3.1 can support *any* simulation system.

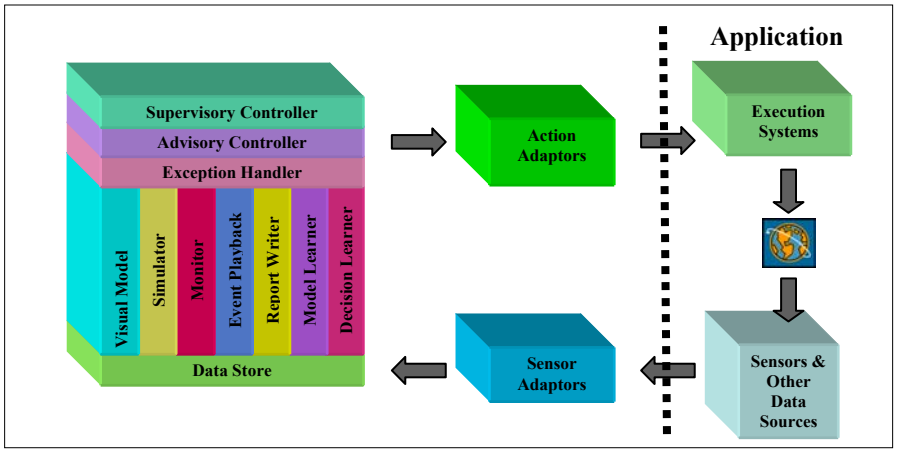

Figure 4 LDI Oasys 3.0 Conceptual Architecture

Conclusions and Future Work

Real-time Simulation-Based LookAhead (SLA) makes better use of time and takes into account the *future* impact of each decision. The more time available to make the decision, the better the decision. Greater decision-making time allows a deeper lookahead, which better gauges the

long-term impact of each decision. Simulation-based decision-making also incorporates a model of uncertainty—decisions that might have uncertain effects. In contrast, it is difficult to model uncertainty with fixed rules. Finally, simulation-based decision-making operates from first principles, which means it can take into account complex interactions among resources. All of these result in increased performance for UAVs.

Although UAVs are poised to offer significant change in the way the military conducts operations, the pilots who fly the UAVs are often overwhelmed with sensory information and are unable to receive the sensory cues that are normally available to an onboard pilot. Automation has been touted as a solution, but this can result in the pilot losing situational awareness and becoming detached. What is needed is an appropriate partnership between the pilot and an automated assistant—one where the pilot choose an appropriate level of interaction and the automated assistant adapts to the pilot's needs and not the other way around. This keeps the human 'in the loop' for increased situational awareness, better division of labor, and better decision-making.

We are currently developing a new multi-level UAV control system built around an automated adaptive assistant. This system will be affordable, scale to a large number of UAVs, is real-time, adapt over time. Moreover it will be in interoperable, reusable, fault tolerant, and error tolerant, and will handle uncertainty.

As an associate system it might offer suggestions for new altitude, headings, and speed; new sensor modes to cover an area; tactics to minimize danger and maximize probability of completing the mission. Or, it might automatically make such choices, under pilot approval, when rapid real-time response is needed. Of course responses are all dependent on the appropriate level chosen by the pilot.

We plan on exploring three broad levels of automation: *manual, semi-automatic*, and *automatic*. The purpose of the *manual* level is to make decisions requiring expertise, past experience, gut-feelings, which the machine is incapable of. The purpose of the *semi-automatic* level is to make decisions that require both the expertise of a human and the calculating ability and speed of a machine.

The purpose of the *automatic* level is to make decisions that require the machine's special capabilities of speed and calculating ability or when the human operator is overwhelmed with information, cannot respond in time, or is information processing is required that is beyond human cognitive abilities.

Ultimately, the human is in charge of deciding which level is appropriate and once done, can be undone at any time. However, the important point of these levels is that sometimes the machine needs help and sometimes the human needs help.

Biography

Dr. Armand Prieditis is CEO and President of LDI. His work centers on real-time decision-making. **Dr. Mukesh Dalal** is Executive Vice-President of LDI. His work centers on business applications of real-time decision-making. The rest of the co-authors are senior software developers at LDI.