

# Position Paper on Scalability of Evolutionary Computation

Arthurine Breckenridge, Mark Boslough, and Michael Peters

Sandia National Laboratories, New Mexico  
PO Box 5800 MS 1004, Albuquerque, NM 87185  
{arbreck, mbboslo, [mpeters](mailto:mpeters@sandia.gov)}@sandia.gov

**Abstract.** We have been researching methods developed for designing behaviors and a directed cyclic graph for an optimized behavior selection mechanism for autonomous mobile robots to use in unoccupied aerial vehicles (UAVs) performing swarm based automatic target recognition. Our work is based on managing *level of behavior*, where behavior algorithms that are initially developed using evolutionary computing methods in a relatively low-fidelity, disembodied modeling environment can be migrated to high-level, dynamic, complex embodied applications. We will demonstrate our concept using *adaptive waypoints*, which allow navigation behaviors to be ported among autonomous mobile robots with incremental adaptation, different degrees of embodiment, and staged optimization. We believe our evolutionary algorithms scale to provide useful mission goal tasks.

## 1. Introduction

Advances in aeronautical and electronic engineering have enabled the production of low cost unoccupied aerial vehicles (UAVs). A person controls one, or perhaps a few, UAVs. Sandia National Laboratories New Mexico (SNL) wants to deploy emergent swarms of simple yet “smart” UAV as mobile agents that perform automatic target recognition (ATR) over a wide area to achieve a strategic goal (e.g., border protection). As entity groups, or swarms, attain large size (hundreds to thousands), the number of potential character interactions scales exponentially and provides both a greater opportunity to coordinate and a costlier penalty for interference. Having centralized command and control then becomes impractical.

The problem of autonomy for UAVs is not trivial because the noise and uncertainty makes it necessary that the individual UAV be aware of its own local goal performance (i.e., ATR), and change its behavior globally to improve the overall joint-mission measures of effectiveness. Restrictions in available resources in the UAV mean that power, security, communications, UAV characteristics, sensor data, network integration, etc. will be constrained, and effective measures of performance need to be quantified. Also, due to the diversity of physical environments and vehicle (or other entity) locomotion abilities, it is currently difficult to develop navigation control algorithms that generalize across entity types and/or physical settings.

In general, more algorithmic work is available than can be loaded into the current control capabilities of an UAV. The limiting factor is balancing the computational and other resources, whether in a virtual simulation or as a physical swarming entity. We use genetic programming to provide a careful balance of algorithmic behavior code and an optimized behavior selection mechanism to achieve the UAV mission goal that is needed. Our concepts resonate with accepted properties of good engineering design: functional modularity, structural regularity, and hierarchy to achieve the highest level of behaviors supported by the resources.

## 2. Current Situations

The staff of the Intelligent Systems and Robotics and Evolutionary Programming departments at SNL researches, designs, and builds complex robotic systems. We have spent three years in a major project called “Graduated Embodiment of Sophisticated Agent Evolution and Optimization” for deploying quality autonomous robots. SNL’s work is based on managing *level of behavior*, where behavior algorithms that are initially developed using evolutionary computing methods in a relatively low-fidelity, disembodied compute environment can be migrated into high-level, complex, dynamic and embodied simulation environment. This work is an extension of decades of work at SNL in robotics, simulation and animation. SNL research has developed a paradigm for hybrid engineering (e.g., combining planning or deliberate with behavior-based or reactive behaviors) that replaces current control systems and centralization with emergence and distributed-ness for autonomous mobile robots. SNL’s test scenario was specific to goal-seeking navigation. The framework with appropriate mission goal algorithms can be used for applications in numerous areas for civil and military markets.

The initial research was to address scaling issues and bottlenecks seen in traditional individual robotic behavior selection mechanisms that when implementing robotic swarms did not scale. SNL researchers chose to use genetic programming (GP) methods to develop robust autonomous robotic behaviors because this method has been demonstrated to work for “proof of principle” problems. In his book, *Genetic Programming III*, Koza [3] documents sixteen attributes that are needed for challenging a computer to solve a problem without explicitly programming it. No other methods come as close as GP, which currently unconditionally possesses thirteen of the sixteen attributes. SNL’s researchers felt the last three attributes: wide applicability, scalability, and competitive with human-produced results were possibly the most important for our application. We felt scaling was the crux of success of evolutionary methods, and cannot be taken for granted, as the history of AI has taught us more than once. [1]

### 3. Position Advocated

We apply “biomimetic behavior” engineering to UAVs for the tasks of trying to figure out where you are, where you are going, and how to get there. Navigation, positioning, and path planning are crucial to virtually every activity undertaken by animals and humans, yet the process often seems impossible to describe, model, or prescribe. In the simplest examples, the goals and rules of motion allow for simple behavior algorithms that can be turned into human-generated computer programs that can control human-created systems. But such programs are brittle and prone to failure if the system does not behave exactly as anticipated. The real world is full of contingencies, unexpected events, multiple (and often competing) goals, nonlinear responses, feedbacks, noise, and complex interactions across a wide range of time scales and levels of organization. Despite the changing environment, animals have evolved the ability to cope, prosper, and multiply in such a world. Their robust behaviors give them the ability to navigate and “plan” their path of motion in order to migrate, search for food, return home, find mates, and avoid predators. They are successful at these goals while simultaneously tending to lower-level tasks, adapting to changes in their environment, dealing with unfamiliar situations, and ignoring irrelevant information.

SNL’s biomimetic behavior methodology uses *incremental adaptation, graduated embodiment, and staged optimization* to address differing levels of behavior. SNL introduces a hybrid artificial intelligence (AI) approach that has many advantages in operating in a complex, dynamic environment. Incremental adaptation creates a structural separation of behavior allowing evolution to reuse modules as high-level building blocks. Graduated embodiment allows interaction with the environment or other individuals, maintaining short individual descriptions but leading to complex collective behaviors. Staged optimization is the iterative evolution of functional units. Our first experimental results of layering these three techniques has resulted in both an individual entity and a group of robots with a robust interface between navigation control algorithms and the specific mission task on which they operate. Such navigation strategies, as move-to-goal, are scalable since they are adjusted to suit the degrees of freedom (DoF) capabilities of each vehicle and its environment. The level of behavior paradigm allows scaling to exactly match hardware in the UAV with the maximum functionality possible.

#### *Incremental Adaptation*

Modular-functional decomposition is a fundamental tenet of the field of computer science. We build modular-functional decomposition “building blocks” that are either computer generated GP or a hand-coded (i.e., 2D or 3D motion planner, autopilot). Each building block has deliberative behaviors when dealing with its environment via input and output connectors. Each building block is incrementally adaptive for level of fidelity. Incremental adaptation is not a simple combination of building blocks; it is an enhancement an existing building block. Individual components are black

boxes encapsulating control knowledge that are algorithms provided from biomechanics, robotic, animation or any field of literature. The components are as simple as complex as needed. For example, a module could initially be quite minimal but enhanced as coding permits. Or, the opposite, functionality is restricted for real-time performance. Or, in the GP module, successive generations can pre-select more sophisticated challenges as well as more refined solutions. In our adaptive waypoint example, the module consists of two-layer hierarchy of a real-time 2D motion planner that provides a fast waypoint for ANY UAV and a trajectory smoother for a SPECIFIC UAV. The results are put into the UAV's flight dynamics model.

#### *Graduated Embodiment*

Modular-functional decomposition is certainly a legitimate architectural principle; it is hardly the only one possible. Reactive robotics has tended towards layered architecture with modifying functionality rather than to the "black-box" organization. We build interacting "building blocks" that can be either computer generated GP or a hand-coded (i.e., collision detection, threat avoidance). An interacting "building block" is one in which information is continually being sensed and behavior is continually being generated. Crucial to the distinction between modular-functional decomposition "building block" and an interaction "building block" is the idea that interactive systems are always operating concurrently/embodyed with their environment. Traditional computer procedures work from input at the beginning to output at the end; they are blissfully ignorant of their environment while executing. In contrast, an interactive (sub-) system is always responding to its environment. Each building block has reactive behaviors. Each building block is graduated through increasing levels of fidelity. With complete switching between two distinct "building blocks", graduated embodiment simply adjusts the number of adaptable states and the scope of their perturbation with the environment in a time-varying constraint. We apply a specific definition of embodiment and degree of embodiment [4] to quantify our interactions. For example, many degrees of freedom bear little relationship to the entity's ability to navigate and we can use the same coarse collision detection independent of the UAV specifications. If and only if a coarse collision is detected, a finer grain algorithm will be deployed. DoFs can also be eliminated when UAVs move in unison with others or swarms. Other DoFs may need to be added to accommodate a robotic swarm.

#### *Staged Optimization*

Our research concentrates on how the resulting solutions, either reactive or deliberate, may be integrated to yield composite controllers with significant broader functionality. We build a "behavior graph" or a vertical and horizontal hierarchical composable control structure to manage the interaction between the modular-functional decomposition and the interacting building blocks. Our graph has provision for static parameters (defined as input/output functions), dynamic parameters (defined as a time-varying "instantaneous" value which maps observation or sensations (input) to reactions (output)), and/or pointers to code containing

algorithms for modular-functional decomposition and/or interacting building blocks. Security is built into the system where each behavior graph value can have read or write privileges to owner, group and world within the UAV and via network access. The behavior graph is not a simple directed acyclic graph such as a scene graph in graphics. It allows for feedback loops. Traversing the graph is based on preconditions, post-conditions, and expected performance that have been optimized to resolve ambiguous transitions between behaviors. The graph also has provisions for level of fidelity. Each level of behavior of the building blocks can be switched on or off or varying levels of fidelity in the traditional style or cross-pollinated in the genetic programming style. Rather than combining all building blocks, staged optimization allows high-level behaviors to be evolved before progressing to details. We feel that staged optimization is the key to scaling of the GPs.

The GP code generates an output file called the Level of Behavior (*LOB*) graph. The data structure of the graph contains the algorithm with the appropriate multiple arrays to keep track of state variables (for high level situational awareness and decisions), calculated integers (for policy table flags), registers (genetically calculated variables) and pointers (for multiple decision trees that depend on policy flags). Our LOB graph differs from Finite State Machines in the diversity of the terminal nodes. By generalizing the GP, behaviors for any environmental rules can be developed with the same code, from the most basics (1 DoF, discrete space) to advance (6 or more DoF, continuous space, full aerodynamics, turbulent boundary layer). The LOB graph, C++ object for the individual UAV, and C++ algorithm code for either the incremental adaptation or graduated embodiment is shared by the GP and modeling and simulation code.

#### *Swarming Behavior Scenario Conceptual Model*

The ability for a UAV group to swarm is actually a composite of numerous behaviors from basic navigation to performing an actual goal mission. In our simplified example for illustration, we show six layers of composite behaviors. The behaviors (Figure 1) are not meant to be taxonomy for autonomous vehicles but a starting example of what is needed to perform a simple scenario mission in real-time and the diversity of fidelity available in the algorithmic community. Sub-behaviors may also be identified as the modeling and simulation progresses. Conditions have been reduced to simple yes or no for illustrative purposes. In reality, the LOB graph would be hard to illustrate for even this adaptive waypoint scenario.

Layer 1: Launch the UAV-- The behaviors can range from fly, un-tethered or minimally human-assisted illustrating the flight phases of a model flight trajectory (e.g., takeoff roll, takeoff, initial climb, on-path climb, cruise climb, initial descent, approach descent, final descent, and landing) to air dropped from another vehicle or sling-shot from a launching mechanism.

Layer 2: Validation of Hardware and Software-- The behaviors can range from preloaded task information that has been verified prior to launch to dynamically loaded task information where task is verified and validated in motion such as checking sensor (i.e., camera) and GPS locations (i.e., first waypoint).

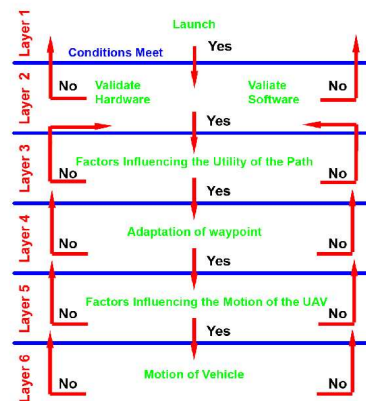
Layer 3: Factors Influencing the Path Utility-- Some constraints to consider are:

1. Collision detection – The behaviors can range from avoiding collisions during all real-time behaviors with high-fidelity exact shape detection based on vision sensors to pre-determined static rough shape avoidance with no dynamic assistance or look ahead capability of a terrain following radar.
2. Region Mapping – The behaviors can range from pre-determined known obstacle mapping to dynamic discovery.
3. Goal Mission – In this example, the behavior is ATR ranging from multiple, dynamic targets with adjustments for more exploration to single static targets, report, and return-to-base. Data collection or communication techniques to determine targets add further complexities to the behavior.
4. Threat Avoidance – The behaviors range from radar threat to line of sight.
5. Collective Navigation -- The behaviors range from thousands of UAVs using particle-in-cell (PIC) codes performing synchronized arrival over diverse environments to two UAVs performing bi-directional communications like a smaller vehicle is deployed from a mother ship. (Note: PIC is a special instance of particle simulation code where forces are interpolated onto particles from a mesh to improve efficiency. Forces can be environmental, behavioral, or directed.)
6. Other additional constraints can be added when coded. The above items are examples only to illustrate the behaviors get diverse quickly.

Layer 4: Adaptation of Waypoint-- The behaviors range from recalculation of next waypoint to read next waypoint from queue.

Layer 5. Factors Influencing the UAV’s Motion-- Some constraints to consider are:

Fig. 1. Adaptive Waypoint Scenario



1. Vehicle movement – The behaviors range from full flight dynamics models including stopping, backwards, hovering to minimal one step forward motion.
2. Vehicle curvature – The behaviors range from full weighted b-spline algorithms for smoothness of trajectory based on parameters such as maximum turn rate, maximum climb rate, degrees of freedom, aggregate altitude along mission path to no curvature allowed.
3. Vehicle sensors requirements – The behaviors range from multiple sensors with multiple perturbations with environment to no sensor.
4. Vehicle modes of effectiveness -- The behaviors range from high dependence on swarm of UAVs to single individual reliability parameters based on travel, fuel limitations, etc.
5. Vehicle communication requirements – The behaviors range from large transfers of video data in harsh environment conditions to minimal “I’m alive” transmissions.
6. Other additional constraints can be added when coded. The above items are examples only to illustrate the behaviors get diverse quickly.

Layer 6. Motion of Vehicle-- The final behavior in the hierarchy is the UAV moves in REAL-TIME.

In our architecture, the LOB graph can be used as a building block. If all parameters are static with strict input/output from a module, it is equivalent to a modular-functional decomposition building block. If the variables are dynamic with time varying instantaneous values, the LOB graph functions as an interacting building block. If multiple building blocks are present, the LOB graph acts as a mechanism for composing behaviors. The LOB files can be concatenated for a human-imposed structure to the graph or computer-generated to a fitness function. In our example, Layer 3 and Layer 5 are NOT simple sequential behaviors. An optimized balance is needed and quickly becomes a factorial problem if done manually. Staged optimization is needed for scaling. For projects involving robot teams, a LOB graph can be optimized for different fitness functions and tasked into different UAVs.

We develop a method of analysis that shows the quality of service metrics of each individual UAV and a swarm. This effort applies to swarm metrics but in general is repeat of the work done at Sandia in 1988 known as the Gustafson-Barsis [2] Law: the size of most problems is scaled up sufficiently, then any required efficiency (A measure of hardware utilization, equal to the ratio of speedup achieved on P processors to P itself.) can be achieved on any number of processors. Or in other words, the size of the goal mission is scaled up sufficiently, and then any required efficiency can be achieved by any number of UAVs operating in a swarm. The relative speedup or accuracy is where the single UAV execution time is divided by the swarm execution time.

#### 4. Conclusions

In a deliberative, logic-based paradigm, mobile robot control is divided into sense, plan, and act phases, with each phase handled respectively. A plan equates to compute in a virtual world. The cycle of sense, plan, and act is repeated at regular and very small time intervals throughout the agent's operational period. This paradigm works best for well-defined, represented problems. Unfortunately, many of the environments where embodied agents might be used (certainly in most real world environments) demand that the agents make decisions with alacrity due to the high rate of environmental change. As environments become more and more dynamic and complex, the problems become more severe, eventually causing the agents based on a sense/plan/act cycle to *become impractical*. We need to show a diverse, distributed, decentralized, and dynamic method for developing UAV control. After developing the individual UAV control genetically, we need to show a useful and self-organizing method for the UAV to become part of a swarm. We use a combination of deliberative and reactive algorithms. We developed a methodology for selecting the algorithms that scale between full fidelity, incremental adaptation and graduated embodiment. All components can use genetic programs and the combination of techniques show that GPs scale.

#### 5. Acknowledgements

Sandia is a multi-program laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000. The Laboratory Directed Research and Development (LDRD) program funded this project.

#### 6. References

1. Brooks, R. A. "A robust layered control system for a mobile robot" IEEE Journal of Robotics and Automation, 2: 14-23 (1986).
2. Gustafson, J. "Re-evaluating Amdahl's Law," CACM, 31, 5, 532-533, (1988).
3. Koza, J., Bennet, F., Andre, D., and Keane, M. "Genetic Programming III: Darwinian Invention and Problem Solving," Morgan Kaufmann, (1999).
4. Newton, A.L., Nehaniv, C.L., and Dautenhahn, K., "The Robot in the Swarm: An Investigation into Agent Embodiment within Virtual Robotic Swarms," Lecture Notes in Computer Science, Vol. 2801. Springer-Verlag, Heidelberg On-line (2004).