Asymmetric Adversary Tactics for Synthetic Training Environments

Brian S. Stensrud, Douglas A. Reece, Nicholas Piegdon
Soar Technology, Inc.
3361 Rouse Road, Suite #175, Orlando, FL 32817
{stensrud, douglas.reece, piegdon}@soartech.com

Annie S. Wu
University of Central Florida
4000 Central Florida Blvd., Orlando, FL 32816
aswu@cs.ucf.edu

ABSTRACT: We describe an approach for dynamically generating asymmetric tactics that can drive adversary behaviors in synthetic training environments. GAMBIT (Genetically Actualized Models of Behavior for Insurgent Tactics) features a genetic algorithm and tactic evaluation engine that - provided a computational specification of a domain and notional representation of the trainee's tactics - will automatically generate a tactic that will be effective given those inputs. That tactic can then be executed using embedded behavior models within a virtual or constructive simulation. GAMBIT-generated tactics can evolve across training exercises by modifying the representation of the trainee's tactics in response to his observed behavior.

1. Introduction

The use of asymmetric strategy and tactics has proven to be an effective threat against U.S.-led forces in the Middle East. While U.S. forces have a significant military advantage over the insurgent adversaries in terms of conventional military force, these adversaries are nonetheless able to further their goals through a variety of means. The adversary strategies and tactics are asymmetric because they do not attempt to counter U.S. strengths with equivalent strengths; instead they attack areas where it is hard for the U.S. to apply its strength. As U.S. forces adapt to eliminate the areas where they are vulnerable to attack, the insurgent forces change their tactics to find and exploit other vulnerable areas. This adaptation is a significant change from adversaries that are assumed to follow a fixed doctrine.

The evolving nature of the threat poses a challenge to standard training practices. Training programs, especially those based on computer simulation scenarios, use a fixed seet of threat tactics. How can any fixed training program train Soldiers to counter an evolving threat? The answer is that expert humans play the role of asymmetric adversaries in training exercises. The human simulation operators can not only employ known, effective adversary tactics, but can creatively improvise new tactics and adapt to exploit the trainee's weaknesses.

The problem with using human role players in training systems is that creative, expert players are not always available. This is particularly true for deployed training systems. What is needed is a way to make computer generated forces in simulations adapt to trainee tactics—providing not only a varied opposing force, but an adversary that exploits weaknesses in trainee tactics. To explore how to meet this need, Soar Technology developed GAMBIT–Genetically Actualized Models of Behavior for Insurgent Tactics. The GAMBIT system features a genetic algorithm (GA)-based tactics generator, capable of searching a complex space of possible actions for novel, effective tactics for adversaries to employ against the trainee. The vision is to generate these tactics automatically, without any intervention from an instructor. The benefit to the warfighter from using GAMBIT is superior training against a more realistic simulated adversary.

The following sections describe the notional training concept that GAMBIT would support, the characteristics of genetic algorithms that make them suitable for this task, and the approach to incorporating GAMBIT into a training architecture. The next section reviews related work in adaptive computer generated forces. The last sections describe a prototype implementation of GAMBIT, the results obtained from test runs, and our vision for future work.

2. Training Concept of Operations

The ultimate goal for the GAMBIT is to provide an tactic generation system that supports training. GAMBIT supports a training approach in which trainees practice a task multiple times. The goal for GAMBIT is not only to provide variety in the threat conditions, but to adapt to the trainee's actions and exploit his weaknesses. For this research we propose a concept of operations (CONOPS) in which a human trainee is given the opportunity to make and execute a series of mission plans. Based on the results of each mission, the trainee can modify their plan and improve the way they execute the plan. Similarly, the insurgent force makes and executes a series of plans against the trainee. The insurgent force plans are generated by GAMBIT.

For this effort, we are focusing on a convoy operations scenario. The trainee in this scenario plays the role of convoy commander, and is responsible for the generation and execution of the convoy plan. The convoy commander and the GAMBIT system generate plans given basic parameters of the mission. The convoy commander's plan would detail a route to his destination, order of march, deployment of support forces, and specific battle drills and operating procedures to use in response to suspicious situations and insurgent attacks. Conversely, the plan generated by GAMBIT would specify what forces to use (e.g. IEDs, sniper teams, RPGs, rifles, etc.), where they are deployed, the sequence/ timing of attack actions and reactions they will use.

The trainee would execute his plan in a simulation environment using some graphical user interface. The adversary forces either execut GAMBIT's plan autonomously or use a human operator to translate the plan description to entity actions.

After the exercise is over, both BLUFOR and OPFOR use observations and lessons learned from the engagement to correct their tactics. Plan modifications on both sides are executed in a second convoy mission exercise, from which a second set of results can be used as input to the planning phase of a third exercise, and so on.

This CONOPS provides the warfighter with a unique opportunity to train against an unpredictable, dynamic OPFOR capable of learning across exercises and adapting to exploit observed weaknesses. Because he cannot predict the OPFOR behavior (from previous observations or otherwise), the trainee cannot 'game' the training system in anticipation of a particular ambush tactic. Rather, he must develop and execute a plan that maximizes the chance of success against an unknown threat. Furthermore, the trainee must be capable of evolving new tactics in response to OPFOR, lest allowing the system to 'game' blue tactics.

3. Genetic Algorithms

First developed by John Holland in the 1960s, genetic algorithms (GAs) are a tool for developing optimized solutions to problem spaces that are too complex to solve analytically (Holland, 1975). GAs operate by creating a large population of random candidate solutions and allowing them to 'evolve' into better ones. In the case of GAMBIT, each candidate solution is a specification of a particular OPFOR tactic. The GA begins assessing the quality of each candidate using a fitness function. In our case, the fitness function is a simple combat simulation. From the initial population, the GA uses a selection mechanism to choose the most fit individuals to "reproduce" and form a second generation of candidate solutions. Individuals with a better fitness value are considered stronger solutions and are, as a result, given preference during selection. The reproduction process uses genetic operators such as crossover and mutation to generate new candidate solutions that are variations and combinations of existing solutions.

Ideally, each generation's population will contain individuals with better fitness values than the population that preceded it. The GA can then be run for an arbitrary number of generations until one individual's fitness has reached a certain threshold. That solution is the output of the algorithm.

The GA is well suited to the tactic creation problem that GAMBIT is trying to solve because, given an appropriately general representation of tactics system, the GA will search for effective solutions without regard for designer expectations or doctrinal bias. While the population should contain conventional and expected solutions, it also may contain unorthodox and unexpected—but effective--solutions. In contrast, tactics generated by humans can be expected to reflect the specific knowledge and bias of a particular human. Humans are less likely to consider the full range of potential solutions in the (large) solution space.

The GA operators for generating new candidate solutions, mutation and crossover, are intended to effectively sample the search space of possible solutions. However, the large search space size presents a significant computational challenge, even for a GA approach. Given the immense variety of tactics that can be represented for a convoy operations domain, a possible consequence is a situation where the GA simply flounders about, incapable of converging to any useful solution. The key to avoiding this situation is to generate a representation of the domain such that the space is large enough to contain unexpected and effective solutions, but not too large to search in a reasonable time. Other efforts (see next section) have successfully found this balance, and our prototype implementation is also reasonably balanced; we thus believe that GAs are a useful approach to the tactic generation problem.

4. GAMBIT System Architecture

Figure 4.1 illustrates the conceptual architecture of a training system using GAMBIT. The flow of information from scenario inputs to tactic execution is highlighted in red. GAMBIT is provided with a set of initial constraints and an initial representation of BLUFOR tactics. The initial constraints provide resource information to the system (e.g. resource cost and availability). The BLUFOR tactics are used by the tactic evaluation module to play against candidate OPFOR tactics.

GAMBIT begins each training iteration by generating a random pool of candidate tactics. Candidate tactics are fed into the tactics evaluation engine. This engine is a simulation that plays each candidate tactic against the given BLUFOR tactics in a simple convoy scenario. The results of the simulation run are scored to produce a fitness value for the candidate tactic. After all candidates are evaluated, the GA selects the ones with the highest fitness values and uses them to generate a new pool of tactics through recombination and mutation. The evaluation process is repeated for these tactics. This process is repeated for a fixed number of generations. When the process is complete, the tactic with the highest fitness value is chosen as the output tactic for GAMBIT.

One a tactic has been selected, the execution phase of the exercise can begin. While the BLUFOR will be controlled by a human operator, the OPFOR tactics can be executed in the training simulation either manually (using a human operator) or through computer controlled behavior models. After each training exercise iteration the observed BLUFOR tactics are encoded by an operator (or, eventually, an automatic recognition module) for GAMBIT to use in the next interation. The system naturally evolves tactics across training itertions if the BLUFOR tactic input changes.

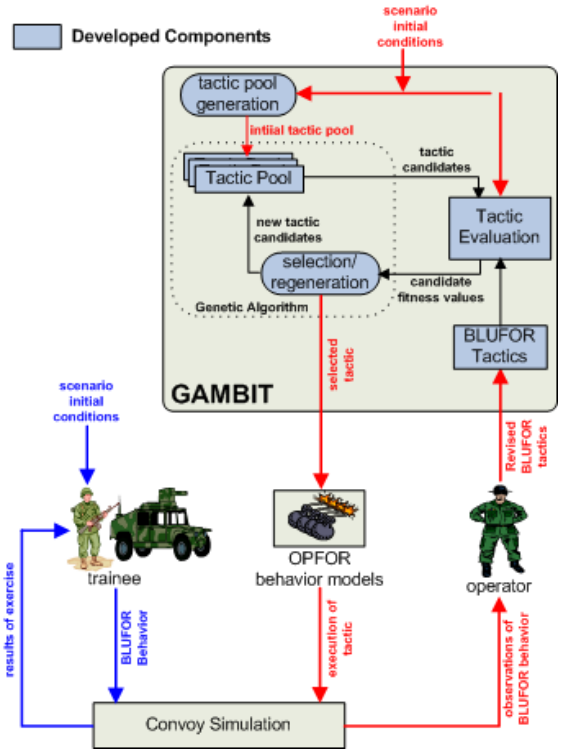

Figure 4.1. GAMBIT Training Architecture

5. Related Work

Numerous researchers have investigated the automatic generation of behavior for synthetic forces in military simulations. One portion of this work involves engineering knowledge into a system that can solve military problems using standard machine planning techniques from Artificial Intelligence (Benoit, Elsaesser et al. 1990). Other work seeks to develop systems that learn unit actions automatically, but use specific training examples to identify correct actions; for example, (Rajput, Karr et al. 1996). Both of these types of efforts have had success in producing rational behavior, but they produce tactics that conform to programmed doctrine. This is usually a desirable effect, but it is not suitable for a system that is required to produce unexpected tactics.

A third category of the automatic behavior generation research uses unsupervised machine

learning instead of learning from doctrinal training examples as above. The goal of this research is often to be able to generate intelligent, robust behavior without having to extract behavior details from a military expert. This approach can produce non-doctrinal behavior--unsatisfactory for many applications (Petty 2001), but exactly what is desired for GAMBIT.

The research in behavior generation with unsupervised learning generally uses Genetic Algorithms. Schultz et al. (1990) used GAs to learn rules that represent tactical plans. These rules can produce sequences of actions that result in a payoff. Schultz was motivated to use unsupervised learning because of a lack of training examples and an intractable domain theory—one requiring simulation to evaluate tactics. Schultz used symbolic condition-action rules, which allows better human understanding of the machine-generated tactics, the potential for supplementing the GA approach with analytical learning, and the ability to seed the learning with human-provided knowledge. The GA found the best rule sets; within a rule set, reinforcement learning adjusted weights of individual rules to improve rule selection when more than one was eligible to fire.

Several research projects have used GAs to generate low-level, physical aspects of entity actions. Fogel et al. (1996) applied evolutionary programming techniques to the generation of behavior in ModSAF, an entity-level combat simulation. This work addressed the control of speed of a vehicle along a route to minimize detection. Tyler et al. (1997) used GAs to find optimal routes for unit travel, considering multiple factors that were too complex to express in a cost function for a traditional path planning algorithm. Kewley and Embrechts (1998) used GAs to find optimal positions of units in a battle area, given an enemy course of action and Hayes and Schlabach (1998) found optimal assignments of units to axes of advance in a battle plan.

The research described above showed that GAs can be used to determine effective behavior parameters for military entities and to find sequences of behavior to achieve mission objectives. Each of these experiments used some designer-provided structure within which the problem solution could be expressed to provide some bounds for the GA solution search; we are also following this approach. Our research in this project extends this earlier work in several respects. First, tactics in GAMBIT include the composition of forces, including both the number of entities and the types of weapons. Also, tactics in GAMBIT involve actions by multiple entities. The number and type of entities can vary. Finally, GAMBIT tactics include, in addition to the composition of forces, the behavior of the forces.

6. Implementation

6.1 GAMBIT's Genetic Algorithm

Our current implementation of the GAMBIT system uses a Proportional Genetic Algorithm (PGA) to evolve OPFOR tactics. The PGA is a variation of the GA that uses a dynamically adaptable representation method (Wu and Garibay 2002). Like a traditional GA, a PGA encodes solutions as linear strings. A PGA, however, uses a multi-character alphabet and encodes information based on the relative amounts of the characters that exist in an individual. The information represented by a PGA individual depends only on what is present on the individual and not on the order in which it is present. As such, the PGA representation is location independent and eliminates issues of positional bias (Eshelman et al. 1989).

Information is encoded in the PGA representation by assigning one or more unique characters to each parameter or component of a solution. The value of that parameter is determined from the relative proportions of the assigned characters of that parameter. As a result, the order of the characters has no effect on the information that is encoded. Characters that exist are expressed and, consequently, interact with other expressed characters. Characters that do not exist are not expressed and simply do not participate in the interactions of the expressed characters. If desired, new characters (and thus, new encoded information) can be added dynamically at any point during a GA run by modifying the genetic operators to include the new characters.

The selection methods and genetic operators that are used in the PGA are similar if not identical to that used in a traditional GA. Selection remains unchanged. Any of the common selection methods may be used in the PGA. The linear representation format allows us to use traditional crossover operators as well. The only operator that required some modification is mutation, because the individual characters that make up an individual are not binary. Mutation is implemented as a random

change to any possible character in the alphabet. The mutation rate defines the probability that each character will be mutated. A character that does undergo mutation is randomly changed to one of the characters in the GA alphabet.

We chose to use the PGA because of its natural flexibility for open-ended problems and its simplicity. We do not know in advance how many weapons will make up a competitive strategy and the goal of our algorithm is to find both the right number and the right combination of weapons to make up a successful strategy. The linear multi-character representation is easy to manipulate using simple genetic operators and easy to decode.

6.2 Representation of BLUFOR tactics

The BLUFOR tactical choices were simplified in our experimental GAMBIT prototype. Since we were focusing on the ambush engagement in this research, we did not address route planning. Further, the convoy composition, march order, speed, and spacing are fixed. There is no opportunity to take actions to detect ambushes.

The convoy commander's decisions in Phase I are limited to commanding the convoy to perform one of several actions:

- Continue move—vehicles continue on the route; used when no damage is done by the ambush, or when the engagement is complete
- Stop in place—a reaction to an ambush
- Move forward out of kill zone
- Move forward or back away from a kill zone
- Attack OPFOR by fire
- Attack OPFOR by fire from standoff positions
- Assault OPFOR positions
- Evacuate casualties

Most of these actions involve different actions by different vehicles; for example, the assault action uses an assault team and fire support vehicles. The actions mostly concern security vehicles, while the transport vehicles remain stopped (either in place or ahead in a safe area). Vehicle recovery and Render Safe Procedures were considered, but were not implemented due to lack of time.

The reactive components of the BLUFOR tactics require trigger conditions. Conditions that were made available for BLUFOR tactics include:

- Initial attack on convoy just occurred
- OPFOR are known to be present nearby (in attack positions)
- There is a disabled vehicle and crew

A complete BLUFOR tactic specification is a set of condition-action rules that specify unit level actions and the conditions that trigger them.

6.3 Representation of OPFOR tactics

As with BLUFOR tactics, OPFOR tactics were simple in the GAMBIT prototype. We do not model mortars, roadblocks, pedestrians, civilian traffic or notable civilian buildings. There is no OPFOR choice about where to set up the ambush—it is notionally in a good place—or where to set up forces relative to the ambush point. All OPFOR are assumed to be in adequate attack positions.The OPFOR are considered to operate in independent teams of one to three people. The teams can be armed with an IED, an RPG, a sniper rifle, or assault rifles. Each team has a choice of when to initiate its attack, and when to break off its attack and withdraw (to safety; no BLUFOR pursuit is allowed). The attack initiation conditions include:

- Make the initial attack on the convoy
- Initiate attack when there is a stopped vehicle.
- Initiate attack when there is a dismounted vehicle crew
- Wait a fixed period of time and then attack

The OPFOR teams continue attacking until they withdraw. The withdrawal conditions include:

- After one attack
- Never
- After 3 attacks
- After a fixed time, whether or not an attack was triggered first.

The complete OPFOR tactic specification consists of the number of teams, and for each team the weapon type, the attack initiation condition, and the withdrawal condition.

6.4 Tactic Evaluation Engine

Because of the complexity of even the simplest of convoy interdiction tactics, it is not sufficient to merely plug tactic parameters into a function to determine its fitness. Rather, tactics must be evaluated through execution against a reactive BLUFOR opponent. To do this, we developed an abstract tactic evaluation engine that takes as input both OPFOR and BLUFOR tactics and runs a simulation of how those tactics might play out in an actual exercise. The results of that simulation then determine the actual fitness of the candidate tactic.

The evaluation engine models OPFOR teams, BLUFOR vehicles, and BLUFOR convoy support crews. Each entity has discrete states. This state includes an entity's current role, whether it is suppressed, whether it is in cover, and a damage level. BLUFOR tactics are represented as a set of rules that trigger different unit-level actions in appropriate conditions. Unit level actions include traveling in convoy, attacking by fire, assaulting, and performing casualty evacuation. Each unit level action is itself defined by a set of rules that describe what individual entities (with different roles) do in that unit behavior. For example, once assigned the CASEVAC evacuation role, a support crew returns to the primary attack location to assist other damaged BLUFOR entities.

OPFOR is represented as a set of small teams. The teams may employ one weapon (in our current implementation)--an IED, rifles, an RPG, or a sniper. The OPFOR tactic defines the number of teams, what type of weapon each employs, the conditions that trigger its attack, and the conditions that trigger its withdrawal.

The simulation consists of a set of discrete locations at which OPFOR and BLUFOR teams can move between and stage attacks. The locations are relative to the primary OPFOR attack location along the convoy route. A vulnerability relationship is defined between each points; for example, BLUFOR teams can travel to location far enough forward or rear of the primary attack location along the road to be safe from OPFOR attack.

. The engine executes by stepping through a series of discrete turns (see Figure 6.1). At each turn, the best matching action for the current OPFOR and BLUFOR tactics is selected for each entity. Actions are then executed simultaneously, along with state changes. The engine will continue to run for a set maximum number of turns or until any of several termination conditions match, indicating that no further state changes can occur.

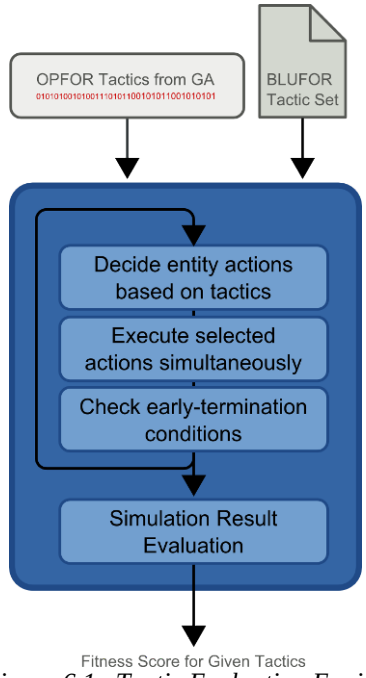

Figure 6.1. Tactic Evaluation Engine

Action selection for OPFOR teams is based on the tactic triple–team weapon type, withdrawal condition, and attack condition–passed in from the GA. If a team's withdrawal condition is met, the withdrawal action will be selected immediately with no further decisions made. Otherwise, if a team is not suppressed and is able to attack, the team's attack condition is evaluated. If met and a suitable BLUFOR target based on the tactic is within range, the attack action is selected. If neither condition is met, the OPFOR team remains in cover.

The simulation models the effects of all combat actions. These effects depend on the weapon type, the target type, the range (in discrete values), whether the target is moving, whether the attacker is damaged, and whether the target is in cover. Effects can include suppression and, for OPFOR, forced withdrawal. The combat models were fabricated to give an approximately correct feel to the combat results, but were not taken from any real data.

Certain actions involve the use of random values to model a more real-world variability in outcomes. The combat model includes probability modifiers

for various conditions. For example, damaged entities are less effective attackers, so a damaged entity would incur a penalty. A modified random number draw is mapped to a results table that describes the state changes that occur. For example, a low attack roll might only suppress the target while a better attack roll could result in both suppressing and damaging the target.

After all actions have been executed and the resulting state changes applied, the simulation checks against several early termination conditions to determine whether the simulation loop can end prematurely and return control to the GA. A termination condition will match when no further changes can be made and the simulation is in its final state. For example, after all OPFOR teams have been destroyed or all BLUFOR teams have reached a point safely ahead of the primary attack location, the simulation can evaluate and return immediately.

The results of the evaluation engine are then used to calculate a fitness for each OPFOR tactic candidate. Fitness values simply reflect the utility of damaging, destroying or delaying the BLUFOR convoy against the cost of the resources used and damage incurred.

7. Results

With our prototype implementation of GAMBIT, we were able to demonstrate several basic results. First, the expected fitness of the tactics generated by GAMBIT increased steadily over time and produced a fairly effective tactic (see Figure7.1). The *expected* fitness is considered because the fitness function includes random outcomes in the combat models, so the results of a tactic could vary significantly from trial to trial. GAMBIT evaluated each tactic ten times and used the average result as the fitness.

The prototype was tested in an experiment in which OPFOR tactics were generated for two different BLUFOR tactics. The different BLUFOR tactics used different responses to the ambush, and employed different battle drills from the list given in Section 6.2. The two BLUFOR tactics are described in Tables 7.1 and 7.2.

Table 7.1 BLUFOR tactic 1

Condition	**Action**
First attack on convoy	React to ambush—stop in place
There is at least one OPFOR team in an *enemy occupied* position, AND there is a disabled vehicle with a crew assigned to it	Attack by fire
No OPFOR in an *enemy occupied* position, AND there is a disabled vehicle with a crew assigned to it	Evacuate casualties
There are no crews assigned to disabled vehicles	Continue convoy

Table 7.2 BLUFOR tactic 2

Condition	**Action**
First attack on convoy	React to ambush—*move forward out of* kill zone
There is at least one OPFOR team in an enemy occupied position, AND there is a disabled vehicle with a crew	Standoff attack by fire
No OPFOR in an enemy occupied position AND there is a disabled vehicle with a crew assigned to it	Evacuate casualties
There are no crews assigned to disabled vehicles	Continue convoy

In each case, GAMBIT produced an OPFOR tactic that resulted in a credible attack on BLUFOR. BLUFOR tactic 1 includes a response to stop (at least briefly) whenever there is an attack. The best OPFOR tactic found included one IED attack on the moving convoy, and then six teams that triggered their attacks when the convoy stopped. This achieved success because the stop triggered the remainder of the attacks. IEDs were the favored weapon because they were able to target additional convoy vehicles driving away with more success than other weapons (as determined by the combat model in the simulation). The BLUFOR tactic 2 response to an attack is to move immediately out of the kill zone; therefore the initial OPFOR attack had to disable a vehicle in order to provide a chance for further attacks. The OPFOR tactic in this case included five IED attacks on the moving convoy, and two RPG follow up attacks. These OPFOR tactics are summarized in Tables 7.3 and 7.4 below.

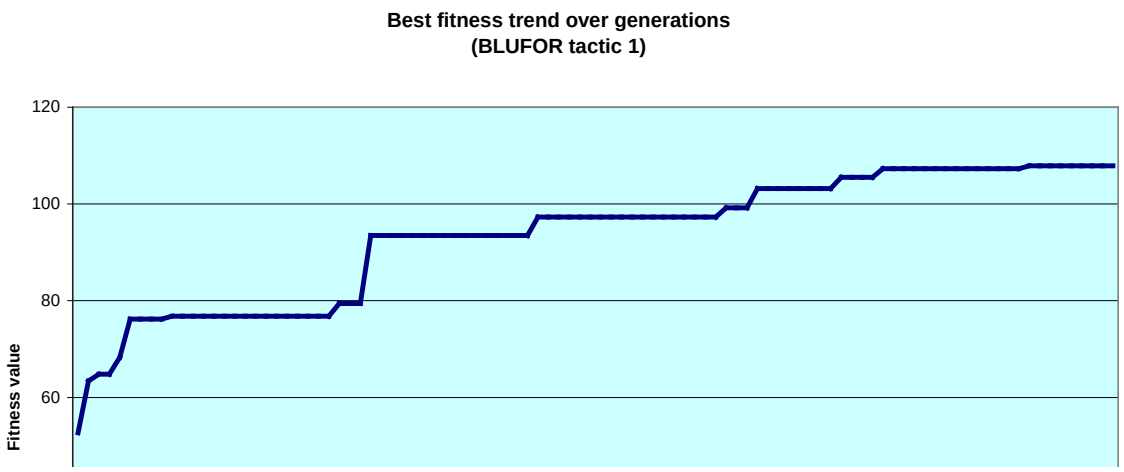

Figure 7.1 Increase of expected fitness of OPFOR tactic over 100 generations

Table 7.3 OPFOR tactic 1

Weapon	Attack condition	Withdraw condition
IED	initial	-
IED	after 3 turns	-
IED	after 3 turns	-
IED	after 3 turns	-
IED	stopped vehicle present	-
IED	stopped vehicle present	-
RPG	stopped vehicle present	Never
RPG	dismounted infantry present	Never
Sniper	stopped vehicle present	Never

Table 7.4. OPFOR tactic 2

Weapon	Attack condition	Withdraw condition
IED	initial	-
IED	initial	-
IED	initial	-
IED	initial	-
IED	initial	-
RPG	stopped vehicle present	immediately
RPG	after 3 turns	Never

The GA in our prototype was run with a population size of 300, a time limit of 200 generations, and a sample size of 10 runs for each tactic. It completed its calculations on an ordinary laptop PC in about 3 minutes.

8. Future Directions

Based on our progress to date, we have a clear set of objectives to achieve going forward to complete a GAMBIT prototype per the requirements of our training CONOPS. First, we intend to continue and expand on our convoy operations domain analysis. We have conducted an analysis of the convoy operations domain and used that information to construct an abstract tactic evaluation system that mimicked elements of that domain. We intend to extend that system to allow for the selection of more robust and detailed tactics, which will require a more detailed analysis.

We also plan to refine our representation of BLUFOR to support trainee tactic evolution across repeated exercises. One of the most important features of the GAMBIT system is the feedback loop from the training simulation that allows GAMBIT to generate new tactics that exploit observed weaknesses in trainee behavior from previous exercises. We plan to refine our current representation of BLUFOR so that a human operator can, through a simple user inter-face or otherwise, encode observed trainee behavior and pass it back so it can be used to evaluate a new population of tactics.

We will also refine our representation of OPFOR and the capabilities of or abstract tactics evaluation engine. As discussed above, we intend to expand

the GAMBIT prototype so that it can generate more robust and detailed OPFOR tactics. This will require the expansion of not only our existing tactic representation but also of our existing tactic evaluation engine. Currently, OPFOR tactics in GAMBIT represent a set of teams, their weapon types, and simple attack and withdrawal conditions for each team. We in-tend to expand on this representation to specify more detailed team types and composite attack and withdrawal conditions, among other parameters. Similarly, we intend to expand our evaluation engine to represent a more complex convoy operations scenario with more decision points and a more robust representation of the players and environment. We will also have to refine our GA design to match this extended representation, examining biases and investigating the relationship between the problem space size and the GA's capability to generate solutions in reasonable time.

Finally, we plan to develop OPFOR behavior models capable of executing GAMBIT tactics and integrate them within an existing training simulation. In order to demonstrate the execution of GAMBIT-generated OPFOR tactics without the use of a human operator, it will be necessary to encode behavior models, within our chosen training simulation environment capable of executing these tactics. We envision these models to be imbued with the ability to perform atomic behaviors (such as IED deployment, retreat, etc.) autonomously, but consume the contents of tactics generated by GAMBIT to determine conditions and sequencing.

9. References

Benoit, J. W., C. Elsaesser, et al. (1990). Planning for Conflict in Multi-Agent Domains. MTR-90W00149, The MITRE Corporation.

Eshelman, L.J., Caruana, R.A. and Schaffer, J.D. (1989). Biases in the crossover landscape. In the *Proceedings of the Third International Conference on Genetic Algorithms,* pp. 10-19.

Fogel, L. J., V. W. Porto, et al. (1996). An Intelligently Interactive Non-Rule-Based Computer Generated Force. *Sixth Conference on Computer Generated Forces and Behavioral Representation*, Orlando, Florida, University of Central Florida.

Hayes, C. C. and J. L. Schlabach (1998). FOX-GA: A Planning Support Tool for Assisting Military Planners in a Dynamic and Uncertain Environment. Technical Report WS-98-02. R. Bergmann and A. Kott, AAAI Press: 21-26.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

Kewley, R. H. and M. J. Embrechts (1998). Fuzzy-Genetic Decision Optimization for Positioning of Military Combat Units. *IEEE International Conference on Systems, Man, and Cyber-netics*, La Jolla, California, IEEE.

Petty, M. D. (2001). Do We Really Want Computer Generated Forces That Learn? *Tenth Conference on Computer Generated Forces and Behavioral Representation*, Norfolk, VA, Simulation Interoperability Standards Organization.

Rajput, S., C. R. Karr, et al. (1996). Learning the Selection of Reactive Behaviors. *Sixth Conference on Computer Generated Forces and Behavioral Representation*, Orlando, Florida, University of Central Florida.

Schultz, A. C. and J. J. Grefenstette (1990). Improving Tactical Plans with Genetic Algorithms. *2nd International IEEE Conference on Tools for Artificial Intelligence*, Washington, D.C., IEEE.

Tyler, J., L. Booker, et al. (1997). Route Planning for Individual Combatants Using Genetic Algorithms. *Proceedings of the Spring Simulation Interoperability Workshop*, Orlando, FL, University of Central Florida.

Wu, A. S. and Garibay, I. (2002). The proportional genetic algorithm: Gene expression in a genetic algorithm. *Journal of Genetic Programming and Evolvable Machines*, 3:2, 157-192.

Author Biographies

BRIAN S. STENSRUD is a Research Scientist and behavior developer at Soar Technology and Principal Investigator on the GAMBIT project. Brian received his Ph.D. in Computer Engineering in 2005 from the University of Central Florida. He also received B.S. degrees in Computer Engineering, Electrical Engineering, and Mathematics from the University of Florida (2001), and an M.S. in Computer Engineering from the University of Central Florida (2003), where he studied evolutionary computation under Dr. Wu. His doctoral dissertation involved the use of a neural network for learning high-level tactical behavior.

DOUGLAS A. REECE is a Senior Scientist at Soar Technology. He has developed agent architectures, reasoning algorithms, behavior models, physical models, and environment representations in individual combatant simulations for the past 15 years. He was the chief architect of the individual combatant and civilian models in the DISAF simulation. His primary research interests are in developing intelligent agents and modeling human behavior for simulations and virtual environments. He has also investigated models of driving behavior and developed the PHAROS traffic simulator to support robot driving research. Dr. Reece has a Ph. D. in Computer Science from Carnegie Mellon University and B. S. and M.S. degrees in Electrical Engineering from Case Western Reserve University.

ANNIE S. WU is an Associate Professor in the School of Electrical Engineering and Computer Science and Director of the Evolutionary Computation Laboratory at the University of Central Florida (UCF). Before joining UCF, she was a National Research Council Postdoctoral Research Associate at the Naval Research Laboratory. She received a Ph.D. in Computer Science and Engineering from the University of Michigan.