Adaptive Battle Agents: Complex Adaptive Combat Models

Kim L. Lim[a], Isaac Mann[a], Ricardo Santos[a], Brad Tobin[a], Matthew J. Berryman[a,c], Alex Ryan[b] and Derek Abbott[a,c]

[a]School of Electrical and Electronic Engineering, The University of Adelaide, SA 5005, Australia.
[b]Land Operations Division, Defence Science and Technology Organisation (DSTO), Edinburgh, SA 5111, Australia.
[c]Centre for Biomedical Engineering, The University of Adelaide, SA 5005, Australia.

ABSTRACT

This work explores emergent behaviour in a complex adaptive system, specifically an agent-based battlefield simulation model. We explore the changes in agent attribute sets through the use of genetic algorithms over a series of battles, with performance measured by a number of different statistics including number of casualties, number of enemy agents killed, and success rate at "capturing the flag". The agents' capabilities include (but are not limited to) manoeuvrability upon the battlefield, formulating, sending, receiving and acting upon messages and attacking enemy agents.

Keywords: complex adaptive systems, agent-based modelling, artificial intelligence, emergent behaviour, genetic algorithms

1. INTRODUCTION

This project incorporates the design, creation and simulation of an agent-based complex adaptive system (CAS) in order to explore emergent behaviour. Specifically, this case will be the modelling of conflict between two groups of agents (soldiers) upon a surface with multiple properties—various terrain types are possible. This project is also focussed upon the changes in the skill/attribute set of the agents between generations, and builds on previous work.

2. BACKGROUND AND SIGNIFICANCE

2.1. Complex Adaptive Systems

Complex Adaptive Systems (CAS) are dynamic networks with a large number of active elements (adaptive agents). The complexity is created by the large amounts of interactions between the system constituents and their environment—in this application the agents and the terrain—with "coherence under change the central enigma."[3] Thus the overall behaviour of the system is the result of the large number of decisions made every moment by the individual agents, with any behaviour in the system arising from competition and cooperation among the individual agents.

According to John Holland,[3] CAS all consist of what is described as the seven basics. These are the four properties (aggregation, non-linearity, flows and diversity) and the three mechanisms (tagging, internal models and building blocks).

The scenarios should also be able to be swiftly designed and simulated with a rapid prototyping capability, which is useful in gathering the information necessary to obtain a more detailed analysis.

2.2. Adaptive Agents

The adaptive agents are able to accurately represent how an individual will interact with others and its environment. The prediction of occurrences and outcomes as experienced by an agent in the simulation becomes extremely difficult, also indicating increased difficultly in optimally specifying an agent's behaviour in advance.

Complex Systems, edited by Axel Bender, Proc. of SPIE
Vol. 6039, 603907, (2006) · 0277-786X/06/$15 · doi: 10.1117/12.638877

2.3. Emergence

Overall, a CAS should show emergent behaviour: "the process of a complex pattern formation from simpler rules. To consider a phenomenon emergent it should generally be unexpected and unpredictable from a lower level description."[4] This term is also used to describe some of the behaviour that should be shown from two of the properties of CAS: aggregation and non-linearity.[1] It describes the complex large-scale behaviour of a system, both from the aggregation of less complex agents and where the agents' behaviour cannot be linearly mapped to the system's behaviour.

2.4. Genetic Algorithms

The two opposing groups of armies that will carry out the conflict in this simulation will also evolve through the use of genetic algorithms: "a search technique to find approximate solutions to a set of feasible solutions—which have been reduced to a discrete set."[4] Genetic algorithms provide for efficient search coverage of the parameters in a simulation, assessing them, judging on their relevance (fitness) and then incorporated into the next step.

3. TERRAIN DESIGN

The battlefield's complexity is increased via the addition of terrain, consisting of buildings, vegetation and elevation.

3.1. Buildings

The buildings are all specified with a minimum size for their length and width, with the placement algorithm using a fractional multiple of this size to improve spacing and layout. Properties include the inability to see, move and shoot through them.

3.2. Vegetation

The creation of the vegetation was done via a fractal generation algorithm using a negative to positive range. The number at the vegetation index of the grid represented a percentage density, with four preset values set. These are:

Bare–0%

Sparse–5%

Medium–10%

Dense–20%

3.3. Elevation

The fractal generation was again used within a negative to positive range. There were no preset height values set, but the number at the elevation index of a specified location represents its altitude.

3.4. Fractal Generation

The algorithm used for the fractal generation of the terrain features is called the Diamond-Square algorithm,[5] and works thus:

1. Begin with a square size equal to the map size
2. Begin with a specified range existing in both the positive and negative region
3. Assign a random number between the specified ranges to each corner of the map

A single iteration consists of a diamond and a square step, where grid points assigned values are treated as either shape (a diamond or a square), their centres seeded with the average of the corners and a random number generated within the pre-defined range. Before each iteration, halve the square size and reduce the random number range according to the formula $H \times 2^{-H}$ to promote self-similarity.

However, the square/map is required to be size 2^{N+1} to accommodate the constant halving of the square.

4. AGENT DESIGN

4.1. Agent Attributes

Each agent has a set of five attributes, representing the capabilities that they possess. The values of these represent their proficiency at each one, and in the first generation of agents are allocated randomly from a pool of fifteen points. After this the attribute values are adjusted (at the creation of each new generation of agents) by the genetic algorithm—they are not changed during a battle. The attributes are used to determine how proficient an agent will be at performing actions from its rule set. Table 4.1 describes these.

Agent attribute	Represents
Communication	The ability and tendency to communicate in a move.
Detection	The ability of the agent to "see" further.
Lethality	The ability of an agent to shoot/kill, and the tendency to do so.
Mobility	The amount of actions an agent can make per step (which actions it makes is dependent on other stats too), and the tendency to choose to move.
Processing	Representing an agent's ability to make good decisions.

Table 1. Agent's Attributes Representations

4.2. Agent Personality, Ability and Actions

The personality of an agent represents the agent's internal value-system as applied to the set of all possible relevant information that the agent must use to select a move or strategy. It is defined by a personality weight vector, which consists of eight different values: The components of the vector specify how individual agents will

Table 2. Agent Personality Weights

Personality Weights	Type of Information
ω_1	Alive friendly agent
ω_2	Alive enemy agent
ω_3	Injured friendly agent
ω_4	Injured enemy agent
ω_5	Friendly flag
ω_6	Enemy flag
ω_7	Changing elevation
ω_8	Changing vegetation

respond to specific kinds of local information within its detection range. They can be both positive and negative; negative weights indicate the agent will tend to move away from that specific entity rather than move towards it. They are also not dependent on the agent's current state—the personality will not change whether it is injured or healthy.

In each battle there are two different types of agents to react to (enemy and friendly), with each agent able to exist in two different states (injured and healthy). There are then four possibilities that an agent can respond to when it detects another, as well as four other pieces of information that can also be acted upon (Table 3). These are incorporated in the environment array, which tracks the agent's current local environment.

The action/ability array of an agent contains the ability at a particular action, as calculated from the attribute weight associated with it. The aptitudes for these are utilised in choosing the action to take, from movement, shooting and communication. The actions and the associated attributes are shown in Table 4.

Table 3. Elements of the agent's environment array.

Index	Environment Description
0	Detected allies
1	Detected enemies
2	Detected injured enemies
3	Enemies in firing range
4	Injured enemies in firing range
5	Messages received
6	Messages sent
7	Shots fired
8	Looking angle

Table 4. Ability vector and associated attributes.

Action	Associated attributes and weight calculation
Fire at injured enemy	Lethality + processing
Movement	mobility
Broadcast "follow me" order	communication
Broadcast "fire at enemy" order	communication - lethality
Execute order "follow me"	processing - detection
Execute order "fire at enemy"	processing
Target enemy	detection + lethality

"Fire at enemy" and "fire at injured enemy" include a limitations in the number of shots that can be fired in one turn, with maximumShotsFired $= \left(\frac{\text{lethality}}{5} + 1\right)$. The enemy must also be in the "field of vision" (Section 4.4). The hit probability of an agent shooting at an enemy is calculated by the difference of their lethality attributes, hitProbability $= (\text{lethality} - \text{enemiesLethality} + 5) \times 9 + 5$. To broadcast the "follow me" order, an agent must have detected an ally but not an enemy. They are restricted to only one message per turn.

To broadcast the "fire at enemy" order, an agent must have detected both an ally and an enemy, but not already sent the message in the same turn. The weight for this is decreased by the agent's lethality; if this attribute is high enough it should take the shot itself.

The order executions occur when the agent has a message in its message buffer, which is checked every move of a turn. The chance of following these orders are governed by attributes—for example, an agent with a high detection has little need to follow another agent as it can adequately detect others by itself.

The "target enemy" action is explained in Section 4.4.

There are also two constraints on agent personalities that have been introduced for specific situations:

Cluster constraint. Prevents the agents from clustering together instead of working to achieve their objectives. Once an agent is surrounded by enough allies it no longer wants to move towards its cohorts. Once the number of allies within its detection range has exceeded that specified by the cluster-constraint the agents will look to move towards other things in it local information.

Minimum distance constraint. This constraint was introduced to introduce a minimum distance between two allies. This is an attempt to give the agent an ability to choose what the best distance to stand from other agents is.

4.3. Agent Movement

The personality weight array governs agent movement by the use of a penalty function. This function measures distances between the agent in question and other agents (be they allies or enemies), flags and local vegetation and elevation. These values are then weighted using the appropriate personality weight and the agent will move towards the co-ordinate with the smallest penalty. If there are two (or more) moves with equally low penalty, in which case the move is chosen randomly between those in question.

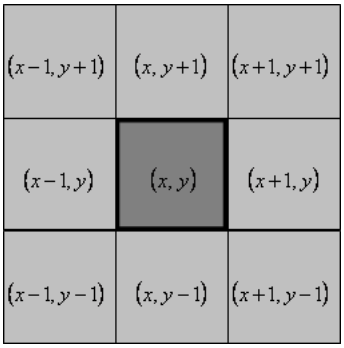

Figure 1. Possible agent positions after one move.

The agents can only move from their current locations by a distance of one in both the x and y dimensions per move. The number of moves an agent may make per step is calculated using its mobility attribute, with a maximum of 8,

$$\text{numberOfMoves} = \frac{\text{mobility}}{2} + 1.$$

Without any goals to move towards (that is, no flags) and a lack of local information, agents will choose random targets and move towards them, only picking another target when the agents reach the first. If an agent encounters

another agent, this will likely change their target depending on the interaction between the two—whether they are mutually friendly or not, or if communication passes between them.

Co-ordinates with a building occupying them have a penalty much larger than any other move can generate to prevent agents moving on them, while vegetation and elevation apply a penalty dependent on the change in density and altitude.

The agents can also get stuck behind buildings when they are directly in the way of reaching an agent's target or flag. To overcome this, the agent ignores the flag for 10 steps and thereby heads toward its target (which is not always the flag), or by resetting the target. In the case of both being on the other side of a building, the conditions on ignoring the flag then resetting the target will take this into account.

4.4. Agent Vision

Each agent has a "field of regard" (approximating peripheral vision), where it can detect the existence of objects, and a "field of vision" (approximating direct vision) which centres upon the agent's target. Agents can only fire on enemy agents when they enter the "field of view".

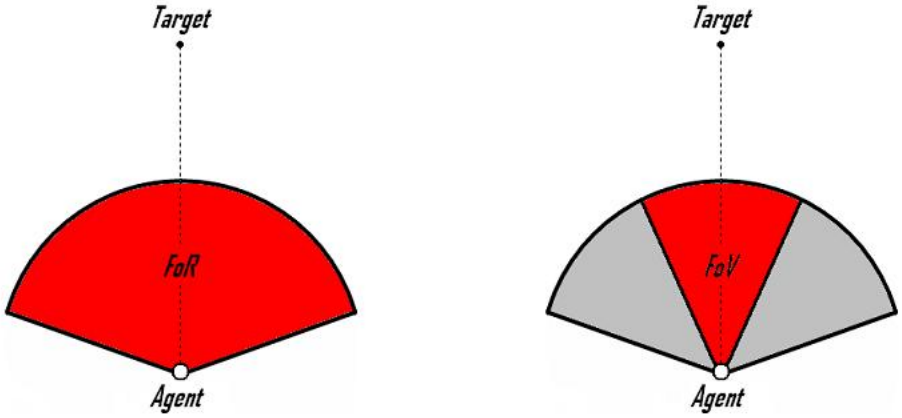

Figure 2. Field of regard and field of vision.

The agent detects its immediate environment using the Moore neighbourhood method, then reducing what it finds to only count the objects within the two fields (by calculating angles). The field's radii are calculated by:

$$\text{VisibilityDistance} = 2 \times \text{Detection} + 1$$

The angles are specified at 150 deg for the "field of regard" and 50 deg for the "field of view". The agent detects its immediate environment using the Moore neighbourhood method, and reduces the detection range to "visibility" $\bigcap$ "field of regard" by calculating angles.

The amount of action upon the battlefield is greatly reduced when agents only shoot at enemies within the "field of view". To overcome this a "target enemy" action was added, such that if an enemy was detected in the "field of regard" but not the "field of view", the agent could move the direction it is facing to have the enemy inside the "field of view".

5. EVOLUTION SYSTEM DESIGN

The evolution system focuses on the change in personalities and attributes of a team until it is able to optimise its performance for the mission objectives and relevant rating criteria.

Chromosomes represent the information being assessed by the genetic algorithm, while the fitness function calculates the fitness of each chromosome by assessing the performance of a team according to their user defined mission statistics.

5.1. Chromosomes

The genetic algorithm has two chromosomes—to represent the personalities and attributes of the agents, with individual agents' personalities/attributes being the genes of said chromosomes.

5.1.1. Personality Chromosome

In the genetic algorithm the i^{th} agent's personality [see Section 4.2 for description] in a team is represented by a string defined by $P_i = g_1, g_2, \ldots, g_{10}$, with g_i as the i^{th} part of the agent's personality. The chromosome is then defined as $C_i = P_1 P_2 \ldots P_N$ with P_i the personality of the i^{th} agent and N the number of agents in a team. The cluster constraints are added upon the end of the chromosome, for the agents to be able to develop to use those values as mentioned in Section 4.2.

5.1.2. Attributes Chromosome

An agent's attributes set (see Section 4.1 for description) is added into the chromosome in the same way, as a string defined by $A_i = g_1, g_2, \ldots, g_5$, with g_i is the i^{th} part of the agent's attribute set. The chromosome is then defined as $C_i = A_1 A_2 \ldots A_N$, with A_i the attribute set of the i^{th} agent in a team and N the number of agents in a team.

5.2. Fitness Function

The system is created with two teams, designated red and blue. The chromosomes of the genetic algorithms are generated from these two teams, representing armies selected from them. The fitness function measures how well each team performs in its mission, based on a number of criteria. In this system, there are five separate weights (ω_i) defined, with each representing the importance of that factor in the mission.

The calculation of the fitness is then $F = \omega_1\beta_1 + \omega_2\beta_2 + \ldots + \omega_5\beta_5$, with β_i the mission formula of the i^{th} weight, $\omega_1 + \omega_2 + \ldots + \omega_5 = 1$ and $0 \leq \omega_i \leq 1$

Table 5. GA Weights and Descriptions

GA Weight	Mission Objective	Description
ω_1	β_1	Time taken to capture the flag.
ω_2	β_2	Minimise friendly casualties.
ω_3	β_3	Maximise enemy casualties.
ω_4	β_4	Minimise the number of enemies close to the flag.

5.2.1. Mission Objective β_1

Objective: minimise time taken to reach the flag.
Fitness formula: $\beta_1 = \frac{t_{f\text{MAX}} - t_f}{t_{f\text{MAX}} - t_{f\text{MIN}}}$

$t_{f\text{MAX}}$: maximum time allowed to capture the flag.
$t_{f\text{MIN}}$: minimum time it takes the closest agent from its starting position to get to the flag.
t_f: actual time the agent took to get to the flag. Only set if an agent gets to the flag and manages to stay there without being killed.

5.2.2. Mission Objective β_2

Objective: minimise the total number of friendly casualties.
Fitness formula: $\beta_2 = \frac{R_T}{R_0}$

R_T: number of remaining allies at time T.
R_0: number of agents at $T = 0$.
T: termination time of the simulation.

5.2.3. Mission Objective β_3

Objective: maximise total number of enemy casualties.
Fitness formula: $\beta_3 = 1 - \left(\frac{R_T}{R_0}\right)$

R_T: number of remaining number of enemy agents at time T.
R_0: number of enemy agents at $T = 0$.
T: termination time of the simulation.

5.2.4. Mission Objective β_4

Objective: minimise the number of enemy agents within a pre-defined distance of the agents own flag (initially set to 10).
Fitness Function: $\beta_4 = \frac{1}{T - t_{\text{MIN}}} \sum_{t=t_{\text{MIN}}}^{T} \left(1 - \frac{R_T(D)}{R_{\text{MAX}}}\right)$

$R_T(D)$: number of enemy agents within a distance D of the flag at time T.
R_{MAX}: maximum possible value of $R_T(D)$.

5.3. Genetic Algorithm

The GA is as follows:

- Read in parameter file for GA data
- If first generation and both team co-evolving then
  - Generate random personalities and attributes for the agents.
- If first generation and one team static then
  - Generate random personality for one team and read in agent files for the other
- For generation=1, maximum number of generations,
- For team=1, maximum number of teams,
  - Decode chromosome,
  - Run one battle simulation
  - Calculate fitness
- Next team
- Rank chromosomes
- Select fittest chromosomes with higher probability
- Perform crossover
- Perform mutation
- Write best chromosomes to file
- Load in new agents
- Next Generation

This process loops until the simulation has done the specified number of generations (or the simulation is interrupted).

6. RESULTS

All tests are run with the following parameters:

- 20 Chromosomes of 20 soldiers
- Battlefield size of 65×65 units
- 40% rank selection probability (used in crossover)

All results have a minimum fitness of zero and a maximum fitness of 100. Each battle is run for 1000 turns unless a team is completely wiped out before then. The surviving team is then the winner and the other the loser. In the case that both teams still have units left after 1000 turns the battle is declared a draw and teams are ranked on their fitness levels. It ensures that battles do not run indefinitely and that there is a limit on how long a batch will take to run.

6.1. Run #1: Minimise Friendly Casualties

The blue team was evolving while the red team was created randomly at each generation. The blue team's mission objective was set to only care about minimising friendly casualties and no terrain was included to ensure the results were only influenced by this weight. Initially at generation 0 the blue team starts off on par with the red team, they are both generated randomly. From here it is expected for blue to gain the upper hand as it has the advantage of evolution. It is also expected that by the end of the 100 generations it should have a well rounded set of armies that can perform well against a diverse set of enemies.

The fitness starts off at a rather low level of twenty but steadily increases after each generation. It is seen that as fitness increases their "aliveFriendlyPersonality" steadily increase along with their lethality and detection attributes. Their "aliveEnemyPersonality" trait decreases along with their communication and processing attributes. It is believed that this shows the blue team attempting to cluster together to try and survive—or "power in numbers" as there were no area weapons implemented in this system.

6.2. Run #2: Mutation Probability 20%

The blue team is an evolving team and the red team was set to be randomly created every generation. The blue team has it's mission objective also set to only care about minimising friendly casualties. Terrain is set to completely flat and mutation probability is set to 20%. The results followed a similar path as that of Run#1, except the fitness increases at a slower rate. Agents' cluster constraints also increase with fitness. Around generation 90 the "aliveFriendlyPersonality" trait takes a drops heavily but was steadily incresing before that point. It is believed that this shows fairly similar results to Run 1 except a higher mutation levels causes fitness levels to increase at a slower rate. This could mean that "good" evolution is much slower when random mutation is involved.

6.3. Run #3: Terrain Impact

To measure the impact of terrain on the performance of a team of agents, the mission objective within the simulation was set to 0.9 for minimising ally casualties and 0.1 for maximising enemy casualties. A set of four simulations were run to determine how varying terrain elements affected the result, with simulations of:

1. No terrain elements
2. All terrain elements
3. 30% vegetation
4. 50% elevation

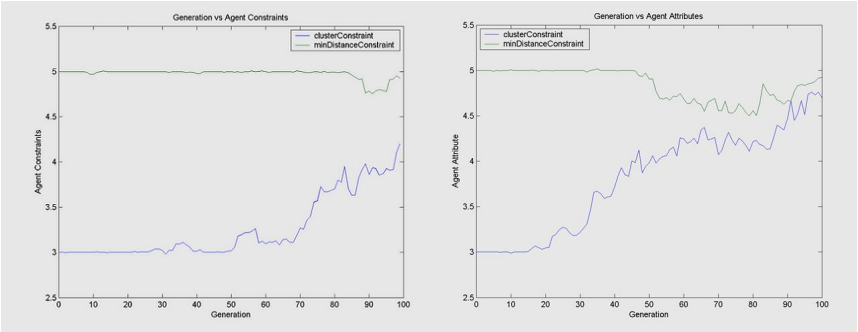

Figure 3. Comparing constraints: all terrain elements, no terrain

6.3.1. Constraints

The minimum distance and cluster constraints showed interesting behaviour in these simulations. With all terrain elements and no terrain the minimum distance stayed around the initial value (5), while the cluster constraint increased—however, the simulations with no terrain began to rise about 40 generations earlier. Thus, agents had realised much faster that strength in numbers is more helpful without the addition of terrain.

With only vegetation, the minimum distance constraint dropped to around 3.5 from 5, implying that to counter the reduction in visibility and killing ability, the agents stayed closer together. All of these results demonstrate that the agents found a large advantage in working closer together as a team than the original settings. A combination of all three terrain elements was required for the slowing increase in the cluster constraint - which didn't occur with either just vegetation or just elevation.

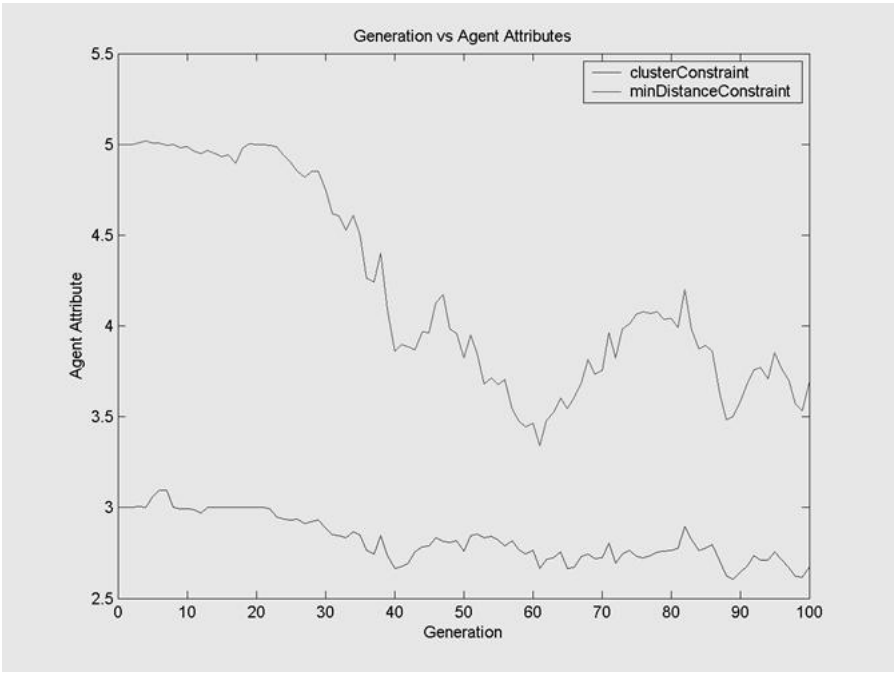

Figure 4. Constraints: 30% vegetation

6.3.2. Personalities

The agents' personality vector showed a strong attraction to alive allies for the case of no terrain, supporting the rise in the cluster constraint. Simulations with terrain proved this was also an important weight in these

instances, however most agents converged to place higher emphasis on moving towards high altitudes to gain a tactical advantage. One interpretation of this could be that teams exposed to battle without terrain were forced to develop alternative tactics to survive, whereas the terrain itself provided enough protection to the evolving team to allow them to specialise in lethality. An interesting point is that with all terrain elements, agents are more likely to move towards injured allies, and less likely to move towards healthy allies than they are in the simulation with no terrain elements.

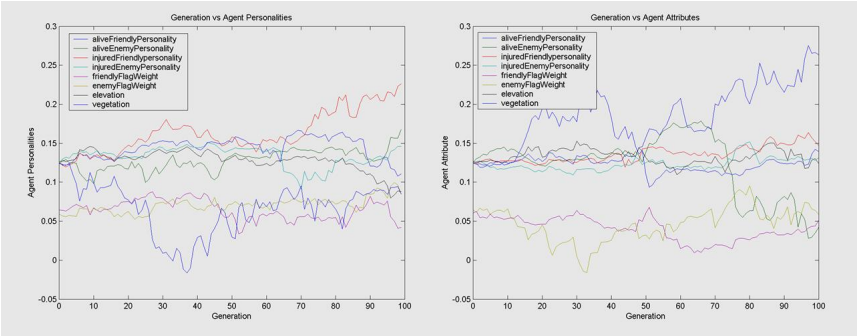

Figure 5. Comparing personalities: all terrain elements, no terrain

6.3.3. Attributes

The attributes of the agents presented similar primary attributes, however detection and mobility were given much higher importance in the case of no terrain. The lack of dependence for detection in this simulation could indicate that the detection skill becomes ineffective when terrain is in use, due to the greatly reduced detection range, irrespective of the agent's ability. This results in the agents selecting other actions to improve their chances of satisfying the simulation goals. The apparent dependency on lethality is greatly enhanced within these simulations as it not only measures the offensive ability of an agent, but also its defense. Because of this, the high lethality attribute in each simulation allows an agent to increase its chances of surviving an attack from an enemy. Mobility is also increased in the simulation with all terrain—this is believed to be due to the agent's lower visibility and buildings, with them required to react quickly (i.e. have the moves available to use) if an enemy agent is detected.

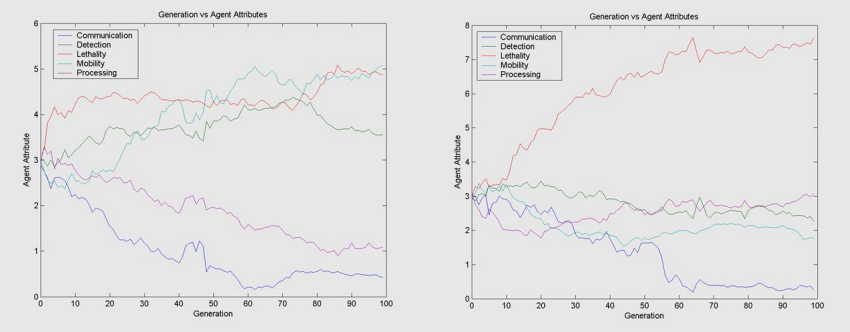

Figure 6. Comparing attributes: all terrain elements, elevation only

7. CONCLUSION

The results from these sets of tests show both complex and adaptive behaviour. The agents evolve to maximise their lethality and detection while maintaining a balance of these two parameters with sufficient mobility to be able to carry out multiple actions in one time step. They exhibit "intelligent" behaviour in areas such as the

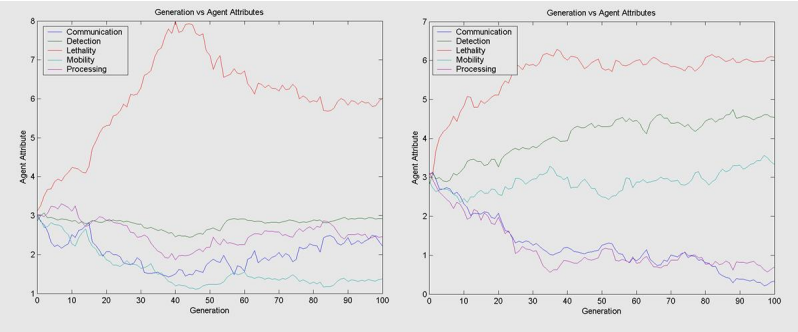

Figure 7. Comparing attributes: vegetation only, no terrain

attribute and personality allocations during evolution—teams attempting to have the best abilities to achieve their objectives—as well as the optimisation of the cluster constraint properties. The agents, for example, stay close together in large groups (especially in denser vegetation) as they achieve a higher number of kills and friendly fire is reduced. A team that has been evolved over 30 generations will almost always defeat a randomly-generated team, but the inclusion of terrain will mean that the agents take longer to reach optimum fitness.

7.1. Future Work

There are several areas in this work that could be improved in the future:

- For the agents, the full use of state changes from alive to injured, and the implications that this might have—such as changes in attributes and personalities. Perhaps (for example) making the agents have less of a reliance on lethality in its actions could be considered. New general agent types could also be created, such as vehicles and aircraft, both with different sets of attributes.
- Changes in the foot soldier agent types could also be made, with attempts to specify certain numbers of infantry types and creating agents with attributes close to what these new types would require. This would also necessitate the introduction of fitness functions assessing much more than numbers of allies and enemies alive and dead and times taken to reach the flags.
- Future work on the terrain could be the ability to save and restore maps, instead of continually generating random terrain, albeit with constant percentages of the terrain properties. Perhaps also being able to specify more detail in the creation of the terrain could be feasible. Other extensions could be the addition of more terrain types such as rivers and roads, and a full implementation of what each change in the terrain will mean for the agents' current attributes, such as slow down movement in vegetation, speed up upon roads.

8. ACKNOWLEDGEMENTS

Funding from the School of Electrical and Electronic Engineering, at The University of Adelaide, is greatly appreciated.

REFERENCES

1. T.J.A. Baker, M. Botting, M.J. Berryman, Alex Ryan, Anne-Marie Grisogono and Derek Abbott, "Adaptive battle agents: emergence in artificial life combat models", *Proc. SPIE: Smart Structures, Devices, and Systems* **5649**, pp. 574-585, Sydney, Australia, Dec. 12-15, 2004.
2. A. Grisogono and A. Ryan, *Desiging Complex Adaptive Systems for Defence*, Defence Science and Technology Organisation, Land Operations Division, Adelaide, 2004.
3. J. Holland, *Hidden Order: How Adaptation Builds Complexity*, Perseus Publishing, Cambridge, Massachusetts, USA, 1996.

4. J. Holland, *Emergence: From Chaos to Order*, Oxford University Press, Oxford, 1998.
5. G. Miller, *The Definition and Rendering of Terrain Maps*, SIGGRAPH 1986 Conference Proceedings, Computer Graphics Volume 20 Book 4, 1986.