Masters Thesis, Department of Computer Sciences, The University of Texas at Austin.

The Thesis Committee for Brian David Boyles
Certifies that this is the approved version of the following thesis:

Evolving Scout Agents for Military Simulations

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor: _____
Risto Miikkulainen

_____
Dana Ballard

Evolving Scout Agents for Military Simulations

by

Brian David Boyles, B.S.

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Computer Science

The University of Texas at Austin

May 2015

Acknowledgements

I would like to thank Professor Risto Miikkulainen, my supervisor, for his patience, advice and invaluable guidance throughout the process of writing this thesis. His class on neural networks inspired me to find ways to apply that knowledge to the confluence of my interest in computers and the military.

Abstract

Evolving Scout Agents for Military Simulations

Brian David Boyles, M.S. Comp. Sci.

The University of Texas at Austin, 2015

Supervisor: Risto Miikkulainen

Simulations play an increasingly significant role in training and preparing the military, particularly in environments with constrained budgets. Unfortunately, in most cases a small number of people must control a large number of simulated vehicles and soldiers. This often leads to micromanagement of computer-controlled forces in order to get them to exhibit the human-like characteristics of an enemy force. This thesis uses Neuroevolution of Augmenting Topologies (NEAT) to train neural networks to perform the role of scouts which analyze the terrain and decide where to place themselves to best observe the enemy forces. The main attribute that the scout agents consider is a vapor flow rate from the enemy starting location to their intended objective, which according to previous studies indicates likely chokepoints along the enemy route. This thesis experiments with different configurations of sensors and fitness functions in order to maximize how much of the enemy team is spotted over the course of the scenario. The results show that these agents perform better than randomly placed scouts and better than scouts deployed using heuristics in many situations, although not consistently so.

Evolutionary optimization of scout agents using vapor flow is thus a promising approach for developing autonomous scout agents in military simulations.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Over the past three decades, computer simulations in the United States Army have become an increasingly important part of maintaining a unit's training and readiness [12][22]. Improvements in modeling and simulation technology allow geographically dispersed military units to train and fight together on a digital battlefield at a fraction of the cost it would take to perform the same training in person [13]. The Army's Project Manager for Combined Arms Tactical Trainers now encompasses simulators for individual soldiers, helicopters, wheeled and tracked vehicles, and artillery [16]. Not only are a wide range of platforms modeled in simulation, but the training focus ranges from individual performance in marksmanship and combat medical care to combined exercises incorporating land and air forces in a battlefield environment. With past budget cuts and a similar cost-constrained environment for the foreseeable future, the importance of computer simulations in the military will continue to increase in the years ahead.

This thesis focuses on simulations that support collective training, which is defined as training that "requires interactions among individuals or organizations to perform tasks that contribute to the unit's training objectives."[9] Exercises in these simulations are always overseen by a number of observer-controller-trainers (OCTs) whose role is to orchestrate the exercise in order to achieve the training objectives set forth by the commander. They can do this through techniques ranging from injecting events into the exercise (e.g. simulating a radio transmission from higher headquarters changing the unit's mission in order to test the unit's flexibility and ability to react to a changing situation) to directly controlling enemy forces in an attack against the friendly lines.

A single OCT normally controls a large number of simulated forces, both friendly and enemy. This practice is particularly common in training conducted by small units (less than 120 soldiers) due to the limited manpower and resources those units have available and the high frequency of small unit training events. While this practice provides excellent economy in terms of manpower and personnel costs, this can also lead to a lower-quality training experience. For example, if there are computer-controlled, OCT-overseen friendly forces in a scenario (to simulate friendly units on the flank of the unit doing the training), the OCT must control their movement, weapons, and make communications with the live unit being trained as if he were the simulated unit's commander. However, if the OCT is controlling several friendly and enemy forces simultaneously he may become fixated on one section of the scenario while letting the other units operate on their own. Focusing too much on one simulated unit or action unfolding in one area of the map can cause the OCT to lose sight of his role in overseeing the exercise as a whole and maximizing the training benefit to the unit.

Given the budgetary and manpower constraints that the U.S. Army operates under, adding additional OCTs is not a realistic solution to overcoming the challenges posed by OCTs controlling large numbers of simulated forces. Instead, the way ahead is to improve the artificial intelligence of the simulated units so that less micromanagement is needed and the OCTs can focus on their primary job of ensuring the exercise as a whole provides the greatest training benefit to the unit, i.e. overseeing how the exercise as a whole is unfolding and making the big picture decisions to maximize the training value of the exercise.

In this thesis, I propose a method for evolving scout agents in a simulated environment using Neuroevolution of Augmenting Topologies (NEAT) [19]. These scout agents will analyze the terrain and determine where to place themselves with the

goal of the friendly team observing enemy forces for as long as possible as they advance across the battlefield. Specifically, the focus of this thesis is to evaluate different sensor configurations and fitness functions for these scout agents and to examine the utility of the gas diffusion method [1] to predict the movement of large military formations.

Chapter 2: Background and Related Work

There are three areas of background information and related work that this thesis is built upon. The first is a general military understanding of reconnaissance operations and the role that scouts play in them. This provides the proper context for the environment that the scout agents will operate in and the support the method of evaluating their performance. The second area is on military terrain analysis and past instances of computer-aided terrain analysis. The final area is a discussion of NEAT and past implementations of this technique.

2.1 Reconnaissance Operations

Reconnaissance is an essential component to every military engagement. If your forces know where the enemy is and the enemy does not know where you are, that allows your forces to choose when and where, or even if, to engage the enemy. In contrast, a friendly force with overwhelming military force is unable to win if they cannot find the enemy. The United States military refers to this as "intelligence, surveillance and reconnaissance" (ISR), which they define as "an enabling operation that integrates and synchronizes all battlefield operating systems to collect and produce relevant information to facilitate the commander's decisionmaking."[4] For ground forces, the primary means of ISR is through scout platoons and cavalry squadrons (which conduct manned reconnaissance from vehicles or dismounted soldiers) though this can also include unmanned aerial surveillance conducted by drones, motion detecting sensors emplaced by ground forces, or information gathered by human-operated helicopters and fixed-wing aircraft.

For defensive operations, reconnaissance falls under the scope of security operations, which are "operations undertaken by a commander to provide early and accurate warning of enemy operations, to provide the force being protected with time and maneuver space within which to react to the enemy, and to develop the situation to allow the commander to effectively use the protected force."[6] The scope of this mission can range from a screening operation, which provides early warning but lacks the combat power to fight any sizeable enemy force, to a covering operation which is fully capable of fighting enemy forces and preventing them from getting close to the main body.

This thesis focuses on modeling a screening operation against an enemy armored force. Therefore, the scout agents in the simulator will not be able to destroy the enemy – their sole focus is to detect the enemy and collectively observe the enemy forces as they pass through the reconnaissance zone so that a notional friendly commander can prepare and organize his forces to fight the enemy. In a screening operation, friendly scouts are typically divided into sections of two vehicles that each go out to establish observation posts (OPs). Depending on the mission requirements, the scout leader can position his OPs in breadth or depth, or designate alternative or fallback positions in order to maintain observation on the enemy throughout the reconnaissance zone.

On the battlefield, scout units are located forward of the main combat units in a zone that can be several kilometers deep. Within this zone, the scouts are given a large degree of freedom in choosing where to establish their OPs. One of the main characteristics of a good OP is that it should offer unobstructed observation of the assigned area or sector, ideally with overlapping coverage from neighboring OPs. However, the second characteristic is just as important: The position should offer effective cover and concealment from the enemy in order to reduce their vulnerability, to include routes to and from the OP [5].

Figure 2.1 shows an example deployment sketch of a screen line on the battlefield with enemy forces moving from north to south. The hills and trees divide the area into three corridors, labeled as "Avenues of Approach" (AAs). The scout platoon leader responsible for this sector has three OPs, which are all placed forward to spot the enemy and determine which AA he will use. The enemy, represented by the diamond, is using AA 2. In response, the scout platoon leader keeps OP 2 in place to observe the enemy and moves his other forces to OP 4 and OP 5, using the terrain to keep the enemy from observing these forces.

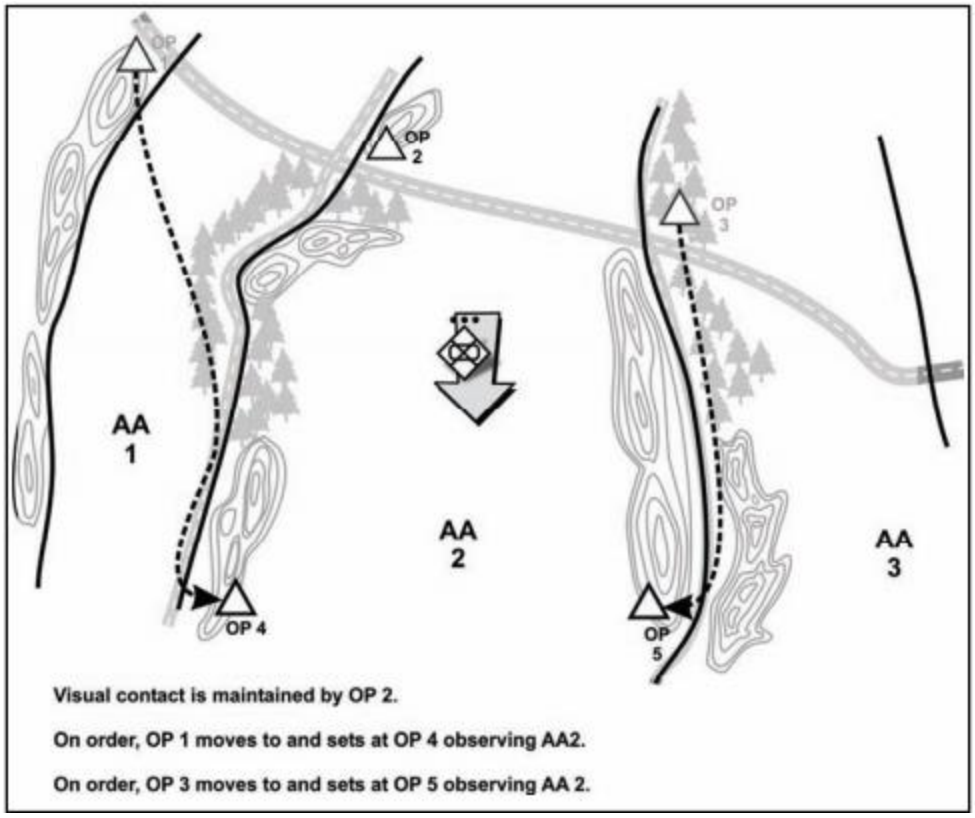


Figure 2.1: Example deployment sketch of three OPs in the reconnaissance zone.

2.2 MILITARY TERRAIN ANALYSIS

Finding and selecting sites for OP emplacement begins in the planning phase, specifically with a terrain analysis of the battlefield. This phase is conducted by a military leader as a part of Intelligence Preparation of the Battlefield (IPB). IPB is a multi-step process which first defines the boundaries of the area of operations, then describes the effects of the terrain, weather, and civil considerations, and finally applies any intelligence available on enemy forces to that terrain to determine where they will be and what they will try to do [5].

The United States Army uses five factors to analyze terrain: Avenues of approach, cover and concealment, observation and fields of fire, obstacles, and key terrain. This thesis will focus on the first three. The starting point is determining the enemy avenues of approach through the terrain. An avenue of approach is "an area that provides sufficient ease of movement and enough width (for dispersion) to allow passage of a force large enough to significantly affect the outcome of the battle."[8] The scouts need to find positions which offer sufficient observation of the avenues of approach, while also finding a position that offers cover and concealment to hide them from the enemy.

Horn and Baxter [10] used the same three factors to produce a terrain analysis tool that located potential tank battle positions to support an attack. They primarily used line-of-sight analysis on digital terrain elevation data to determine locations that could see the enemy forces. Based on this analysis, they created a cost map biased towards routes the enemy could not observe and used Dijkstra's algorithm [2] to find the best route. The user could also add additional constraints, such as a minimum or maximum distance from the objective, or the angle of fire between the assault and support positions to avoid fratricide. Their final product is a terrain analysis tool to assist military leaders in conducting the planning process.

In an earlier work, Richbourg and Olson [17] used a similar approach with line-of-sight calculations and other metrics, but applied to a defensive scenario. In this case, special care is given to identifying concealed routes that the enemy might use to approach the defenses and then trying to find defensive positions that minimize enemy concealment. They use the slope of the terrain as an indicator of whether or not a proposed position is suitable for the occupying unit and attempt to identify larger unit positions by groupings of smaller ones. Like Horn and Baxter, their approach was intended to be used as a planning tool for military leaders to narrow the number of possible defensive positions they would have to consider.

More recently, Mora et al. [15] use a multi-objective ant colony optimization algorithm called CHAC to plan routes for military logistical convoys on a hexagonal map from the computer game "Panzer General." This work builds off their earlier research [14] that first introduced this approach to military route planning. There are two cost maps, one for each objective: The first represents the need to deliver their supplies to the destination quickly and is a standard movement cost map of the area. The second goal is to maximize safety. Enemy units are placed on the map and the locations they can observe are given an associated health cost to reflect the danger of moving through those hexes. In an iteration, the ant agents try to find a complete path from source to destination based on heuristics (the weighted value of speed and safety) and pheromones left by previous ants.

Burgess [1] had a similar motivation to this thesis in that he sought to "reduce the required size of training enablers by exporting some of their current duties to an artificially intelligent system which is able to see digital terrain, recognize militarily likely avenues of approach, and template defensive positions on the terrain…" His approach was different from other works on this topic and is based on Duckham [3], who

proposed that humans tend to choose simplest paths over mathematically optimum paths. As a result, Burgess discovered that a gas diffusion model is very useful for determining avenues of approach for mechanized ground forces. He used approximately 600km^2 of digital terrain elevation data (DTED) covering part of the U.S. military's National Training Center in a scenario in which friendly forces are defending against an enemy attack. The enemy starting positions or anticipated locations are modeled as vapor sources while their objectives are modeled as vapor sinks. The rate at which vapor can transfer between each map cell is determined by the slope of the terrain, which he classified into three categories according to Army doctrine. The map cells are represented as squares 30m across, which exchange up to 25% of the vapor difference with its neighbor.

He noted that upon reaching an equilibrium state, the gradient-vector magnitude is highest at the source, sink, and at chokepoints where the terrain forces enemy units to go through a specific area. Burgess then proposed different methods of using those chokepoints to identify areas of key terrain and possible anchor points to deploy defensive units. These agents would choose defensive locations by scoring prospective locations on their ability to cover the chokepoints by direct fire, proximity of friendly units, and dispersion to reduce vulnerability to artillery fires.

2.3 NEUROEVOLUTION AND NEAT

This thesis has similar foundations as Burgess' paper [1] in that it has a similar motivation and employs the same gas diffusion model to visualize the terrain. The most significant difference is that the scout agents in this thesis are evolved using neuroevolution. Neuroevolution is the artificial evolution of neural networks using an

evolutionary algorithm over a number of generations. In traditional neuroevolution, the topology of the network is fixed and usually consists of the sensor inputs being fully connected to a single hidden layer, which are fully connected to the outputs. While such a network can, in principle, evolve to approximate any continuous function, the search space can be very large and thus require a very long time to train.

A variation of this approach is Neuroevolution of Augmenting Topologies (NEAT) [19]. NEAT begins as a simple, minimalist network whose topology can be augmented by adding or removing links and nodes, therefore keeping the search space small and evolving a network topology that is complex enough to accomplish the desired task but simple enough to learn and develop quickly. NEAT outperforms other neuroevolution methods in traditional benchmarking tests such as double pole balancing [18] and therefore is well suited for the high-dimensional state spaces considered in this thesis.

In NEAT, the links between the network nodes are considered to be connection genes, and the nodes of the network are represented by node genes. These genes are uniquely labeled with an innovation number so that corresponding genes in crossovers can be easily found without requiring a detailed analysis of the network topology. During mutation, network weights can change as in other forms of Neuroevolution, but additional mutations include adding a new connection gene, enabling or disabling genes, or adding a new node. When a new node gene is added, it splits an existing connection gene with the incoming weight to the new node set at 1.0 and the outgoing weight set to the original connection weight. This procedure allows new future innovations from the modified topology but does not immediately affect the functionality of the network. NEAT groups organisms into species according to their network similarities, and agents

compete primarily within their own species which helps maintain diversity in the population.

Some previous applications of NEAT include evolving a roving eye for Go [20] which has a limited field of view of the game board with some sensors providing specific information on the map spaces that the roving eye can see, while others give only a general sense of how many pieces exist outside of its field of view. The network outputs determine where to move the eye and when to place a piece. A similar approach could be used in a scouting scenario in which the scouts have a limited field of view and some sense of where other units are (simulating radio communications) and use that to determine whether to stay in place or find a better observation point.

Another implementation is in the OpenNERO video game [21] in which the user trains neural networks using NEAT in a real-time setting by controlling the fitness rewards and punishments for various behaviors and then competes against other populations in simulated battle. OpenNERO performs NEAT in real-time by evolving and replacing agents in only a portion of the population at a time instead of evolving the entire population at once. This gradual evolution of the population makes the transition from generation to generation less noticeable to a human player while still evolving and improving performance in a relatively short amount of time.

2.4 CONCLUSION

This thesis develops the field of autonomous agents in military simulations by combining Burgess' gas diffusion method to analyze terrain [1] with agents evolved using NEAT [19]. Since this approach has not been tried before, a simulator and

environment was developed to train and evaluate scout agents against non-evolutionary baselines.

Chapter 3: The Bare Bones Military Simulator

In order to train and evaluate scout agents, I developed the Bare Bones Military Simulator (BBMS) which incorporates various terrain types, performs line-of-sight calculations, models the gas diffusion method proposed by Burgess in [1] for modeling military movements across terrain, and uses NEAT [19] for training the agents. The purpose of this simulator is to evaluate the utility of evolving neural networks for scout employment as a proof-of-concept. Therefore it is optimized for running large numbers of experiments at the cost of a simpler representation of the operating environment.

3.1 Basic Functionality

The building blocks for training and testing scout agents are scenario files that the user can create and modify. Each scenario has a hexagonal tile map of arbitrary size consisting of up to three terrain types. Visibility between two hexes that are *n* hexes apart is calculated by drawing a line between them and then checking if any hex along that line blocks the line-of-sight. Table 3.1 shows the impact of terrain type on line-of-sight as well as the movement cost of terrain, which is used to calculate gas diffusion rates. Each scenario also has a maximum viewing distance that reflects the weather and lighting conditions of the scenario.

Each hex can contain up to one unit that can be either friendly or enemy. Friendly units, referred to as scout agents, can represent a manned scout section (on foot or operating a vehicle) or a remote sensor that has been emplaced by some other means. Enemy units represent tanks advancing along a route to their objective. Unit movement paths consist of a series of waypoints the unit will move towards at the rate of one hex per clock tick (0.1 seconds in game time). The simulator supports sub-hex movement to

reflect the actual speed of the vehicles in question, but the simpler movement rate was chosen for these experiments in order to improve the running time. Using realistic movement rates would not change the results. It would only matter if units are moving through trees since it is the only terrain type with a different movement cost from clear terrain. In the actual scenarios, enemy units almost always avoid trees because the enemy units represent mechanized forces that are more vulnerable to enemy forces and travel slower through forested terrain.

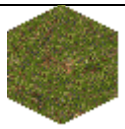
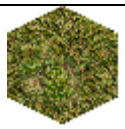
Type	Icon	Line-of-Sight	Movement Cost
Clear		No impact	1
High Grass		Blocks LOS if more than three hexes deep	1
Trees		Blocks LOS to all hexes behind it	9

Table 3.1: Simulated terrain types and characteristics.

Each hex can only hold one unit at a time. Therefore, if an enemy unit tries to move into the same square as a friendly unit, the scout agent is removed from the scenario. This action represents the enemy detecting and destroying the scout, with the assumption that the enemy is unable to detect the scout agent from further distances (through appropriate camouflage and concealment, and in the case of remote sensors, their small size). If the hex is occupied by a unit on the same side, then the moving unit will stay in place and attempt to move during the following tick.

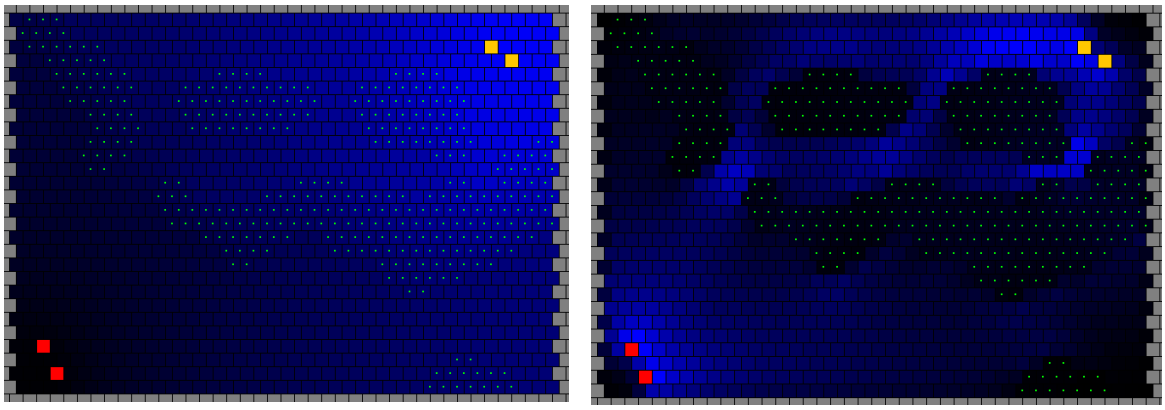
The time flow of the scenario is controlled by a clock that can move at a variable rate. The clock can be adjusted between $1/8^{th}$ normal speed and x96 speed so the user can run experiments quickly. The fitness functions for evaluating agent performance and overall team fitness are unaffected by the clock speed since it only changes the number of ticks that are processed per second.

3.2 GAS DIFFUSION METHOD INTEGRATION

BBMS uses the gas diffusion method developed by Burgess [1] to simulate troop movement. Each hex on the map has an associated gas density attribute, referred to as vapor, which is an integer between 0 and 25500. Since the vapor flow represents the movement of enemy forces from their starting positions to their intended objective, vapor sources on the map represent the enemy assembly area or their anticipated axis of advance. Likewise the vapor sinks represent the objective towards which the enemy units are moving. Sinks empty all vapor in the hex, while sources always replenished vapor in the hex to 25500, with updates occurring every clock tick.

The amount of vapor to transfer to the surrounding six hexes is calculated based on the relative vapor levels in the two hexes and the movement cost of the terrain. The transfer rate is $1/6^{th}$ of the difference in density between the two hexes. This rate was chosen to maximize the amount of vapor that can be transferred while preventing a scenario in which the total vapor transferred out exceeds the supply within the hex (such as a case where a single hex with full vapor density is surrounded by empty hexes). Every clock tick, the transfer amount is calculated for all hexes first, then the vapor is transferred and finally the sources and sinks are updated.

Vapor sources and sinks are placed by the user during the scenario design phase, who then runs the clock until the vapor map reaches an equilibrium state. In an equilibrium state, there is a steady progression of vapor density between the source and sink regardless of terrain, but the important measure is the vapor flow rate from hex to hex. This rate is calculated by summing all vapor flowing into a hex and subtracting all vapor flowing out. Flow vectors can be easily calculated from these rates to determine the direction of flow, but in the experiments only the overall flow rate is used. As with Burgess' experiments, the highest flow rates are found at chokepoints in the terrain and around the source and sink. Figure 3.1 compares the vapor density map (a) and vapor flow rate (b) in a sample scenario. The vapor sources are represented by yellow squares, the sinks by red squares, and trees by green dots. In the vapor density map, the color of each hex is a color gradient that scales from black at zero density to blue at full density. Note how the vapor density changes gradually from source to sink regardless of the terrain type. The vapor flow rate, with the maximum color gradient normalized by the highest flow rate on the map, shows areas of high concentration at the chokepoints between the source and sink.



(a) (b)

Figure 3.1: Vapor density map (a) and vapor flow rate map (b) once an equilibrium has been reached between source and sink in a sample scenario. Vapor density is not useful for visualizing the terrain, but vapor flow rate is high at source, sink, and chokepoints between them.

One limitation mentioned by Burgess [1] is that it takes a long time to reach an equilibrium state. The map Burgess used consisted of 259,200 cells and took hours to reach an equilibrium state. Although the scenario maps in this thesis are smaller by several orders of magnitude and reach an equilibrium state in less time, two new methods that make this process considerably faster have been implemented in this thesis.

The first technique uses a simple distance heuristic to initialize the vapor density level of each hex. Burgess' approach [1] initializes all points on the vapor density map to full density and then lets the vapor sinks and sources reach an equilibrium state. In contrast, the heuristic used in this thesis takes advantage of the fact that, at equilibrium, there is a steady gradient of vapor density between source and sink regardless of terrain. Therefore, this heuristic initializes the vapor density as a function of distance between the nearest source and sink on the map. The advantage of this approach is that all hexes can simultaneously begin moving towards their equilibrium state and they already start close to their final vapor density value. An experiment run on one of the scenarios shows that

this heuristic alone causes a map to reach equilibrium state two and a half times faster than normal (see Table 3.2 for the exact result).

The second technique is to apply a flow rate multiplier so that the vapor transfers between hexes faster. This approach violates the earlier limits on flow rates (therefore making it possible to transfer more vapor out of a hex than exists in it, which in turn can generate significant oscillations in vapor density), but this effect rarely occurs in practice. Therefore, a throttling process adjusts the flow rate multiplier that increases the flow rate by 0.05 every hundred stable ticks up to a maximum of 3.0. Since the hexes that should have the greatest flow rate are the sources and sinks, higher flow rates in ordinary hexes indicate that there may be an oscillation caused by accelerated flow rate. Therefore, if this situation arises, the throttle reduces the flow rate by 0.05 to a minimum of 1.0. Finally, in the event that any hex has negative vapor density, the flow rate is immediately reduced to 1.0.

The throttling method still leads to occasional oscillations in the flow rates on the map, but as it adjusts the flow rate the oscillations dissipate. Furthermore, in spite of the oscillations, throttling reaches an equilibrium state nearly four times faster than normal. Using both methods together increases performance to over eight times that achieved by Burgess in [1].

Technique	Ticks to Equilibrium	Rate Comparison
No prediction or throttling	39647	1.0
Prediction but no throttling	14976	2.65
Throttling but no prediction	10076	3.93
Throttling and prediction	4855	8.17

Table 3.2: Impact of flow rate techniques on the Scenario 41 map. Combining throttling and prediction reaches equilibrium over eight times faster than the standard approach described by Burgess in [10].

On large-scale maps (such as the 30 x 20km map used by Burgess), even using the techniques described above, reaching the vapor equilibrium state will still take some time. However, in the context of this simulator, the vapor flow rates at equilibrium are calculated during the scenario design phase and are not updated in real time. This approach represents the planning conducted prior to the operation, specifically, the terrain analysis of possible enemy avenues of approach.

3.3 NEAT INTEGRATION

BBMS uses the Java version of NEAT (JNEAT), which was written by Ugo Vierucci in 2002 based off of the original NEAT libraries for C++ written by Kenneth Stanley [24]. The package provides support for a large variety of parameters in evolving populations, but only two were varied in this thesis: the overall mutation rate (to include adding nodes and links, enabling or disabling genes, and modifying link weights) and the species drop off age, which penalizes species that persist for too many generations.

The JNEAT population files are saved separately from the scenarios which makes it easy to train a population on multiple maps and to run evaluations progressively.

BBMS also incorporates tools into the simulator to assign organisms from a population to the friendly agents on the map randomly or sequentially. Furthermore, it can get the organism's network output for a selected hex or from the unit's current location.

3.4 SCENARIO DESIGN

To create a new scenario in BBMS, the user creates a blank map or a randomized map of a given size through the user interface. The user then modifies the terrain as needed and adds vapor sources and sinks according to the enemy's starting point and objectives. Once complete, the vapor model is run until an equilibrium state is reached.

The next step is to place enemy units and determine their routes. This process is modeled through "Courses of Actions" (COAs), which represent a unique deployment of enemy units and their associated waypoints in the scenario. Each COA reflects a possible plan of advance by the enemy commander. As many COAs as desired can be added, of which one is randomly chosen during the scenario training iteration. All COAs are currently weighted equally for random selection and testing evaluation. In reality, some COAs are far more likely to be chosen than others and therefore the overall performance of a JNEAT population on a map (in all of its COAs) does not reflect its likely performance against human opponents.

Third, the user must specify where the "friendly zone" and "enemy zone" begin. The friendly zone represents the area in which combat units behind the scouts and support troops reside. Scout agents will never be deployed within this zone, and once all enemy units have reached the friendly zone, the scenario iteration ends. Since enemy units may "wait" in the friendly zone until other units arrive (since enemy units cannot advance off the map edge), enemy units in the friendly zone do not count for or against

team performance or individual fitness. The "enemy zone" represents the enemy's deployment area and thus friendly scouts cannot be placed here. All hexes between the friendly and enemy zone is considered to be the "recon zone" in which friendly scouts may deploy freely.

No friendly units are added during the design process. They are automatically added and placed during scenario training and testing based on user parameters. Once the scenario is complete and saved, the next step is to train a population on it by running the scenario.

3.5 SCENARIO EXECUTION

When loading a scenario for training, the user can either load a pre-existing JNEAT population file or generate a custom population based on parameters for the number of starting sensors, hidden nodes, and starting link probabilities. After specifying various parameters for the scenario, the user will either generate or load the population. Once the scenario is unpaused, it will run until all enemy units have reached the friendly zone, at which point another COA is chosen randomly and the scenario starts again.

To account for the probabilistic nature of neural networks and the chosen COA, each organism's fitness can be averaged over the course of several iterations, so that a single bad iteration will not remove an otherwise successful agent. After all organisms have had the minimum number of iterations, the lower half of the population in terms of fitness are removed and replaced by mutations of the higher-performing survivors. This new epoch is saved as a separate population file and the training continues again.

The user also specifies the percent of the population that should participate in a given iteration. Organisms are chosen sequentially to ensure all agents have at least the

minimum desired number of training iterations desired. This approach significantly impacts the performance of the team: Having too many agents will saturate the map, guaranteeing a very high team performance rate regardless of individual performance.

3.6 TEST POPULATION

During training, the user specifies a directory for saving the population files and the statistics from the training iterations. Each epoch is saved as a separate population file so that performance over time can be evaluated in the testing phase. During testing, the user chooses which directory to read JNEAT populations from and specifies the number of scouts that will participate in each iteration. The user also determines how many runs will be done on each test case (i.e. COA) and the epoch interval for evaluating a subset of the trained iterations (e.g. an epoch interval of five will evaluate every fifth epoch of the population).

Once the target population is loaded, the simulator will automatically go through each COA on the maps specified in the program. For each iteration, the requisite number of organisms are randomly drawn from the population. Once sufficient number of tests have been run for a given COA, the next COA (or the next map) is selected and the process continues. As with training, the summarized and detailed results are saved to text files for easy data analysis.

The next chapter lays out the specific parameters and scenarios that are used to train and evaluate the scout agents in this thesis. It also discusses a variety of organism types and fitness functions that will be compared to non-evolutionary baseline techniques for emplacing the scouts.

Chapter 4: Experimental Setup

The goal of these experiments is to evolve scout agents who are collectively able to observe enemy units as they pass through the reconnaissance zone. The main variables in the experimental setup are two evaluation metrics, three organism types, and four fitness functions which are trained and evaluated across four scenarios. The goal is for the evolved scout agents to outperform two simple methods of emplacing scouts which do not use neural networks.

4.1 Evaluation Metrics

The main method for evaluating team performance is based on how long various enemy units were seen during the course of a given iteration. Every tick, the team gets a point for every enemy unit that it can see. This score is normalized at the end of the test by the total number of possible spots (i.e. *number of enemy units * time spent in the recon/enemy zone*). The maximum performance of a team is highly dependent on the map, number of friendly scouts available, and the enemy COA so direct comparisons of team performance should not be made with other scenarios or within scenarios of varying environment parameters (e.g. number of scouts and maximum visibility range).

A secondary evaluation metric is the number of scout agents that were destroyed over the course of the scenario. Scouts are only destroyed when an enemy unit occupies the same hex as a friendly scout. The individual fitness level of the destroyed unit is inherently reduced because once destroyed it can no longer spot any units. To further discourage scout agents from placing themselves in areas that might be overrun, the "death penalty factor" parameter is multiplied by the agent's fitness if it is destroyed (e.g.

if an agent with fitness 0.8 is destroyed and the death penalty factor is 0.1, its final fitness is 0.08).

This secondary metric is a gross oversimplification of reality, since the enemy might be able to spot the scout from longer range and with the exception of certain remote sensors, the scout agent would be able to move or seek better concealment as enemy units approach. Modeling counter-detection and friendly agent destruction is an area for much improvement in future work.

4.2 Organism Types

All organisms modeled in BBMS are neural networks evolved using JNEAT. The sensors are the inputs to the network and they measure the equilibrium vapor flow rate on the scenario map. The number and configuration hidden nodes vary from organism to organism, but all use the sigmoid activation function to calculate their output. There is a single output for each organism which represents its evaluation of a prospective OP location. If the network output is greater than or equal to 0.75 it will move to that location and remain in place for the rest of the scenario.

4.2.1 Single-Sensor Model

The simplest type of network consists of a single sensor that sums the total vapor flow rate in all of the hexes it can observe from a prospective hex. This value is normalized by the maximum possible flow rate observable from any hex on the map. The rationale behind this approach is that the vapor flow rate indicates the likelihood of an enemy unit moving through a particular hex, so therefore the sum of vapor flow rates

visible from a hex should measure the overall likelihood of observing the enemy from that position.

4.2.2 Dual-Sensor Models

This thesis considers two types of dual-sensor models. The inspiration for the first model arose due to the high death rate of scouts in one of the training scenarios. In the scenario, a three-hex wide corridor penetrated an otherwise thick forest. The greedy approach to getting spots would lead scout agents to position themselves in the middle of the corridor, which would inevitably lead to their destruction (see Section 4.5.1 for details on this scenario). With only one sensor to detect flow rates, there was no way for the organism to learn what a "safe" spot with good observation is compared to a "dangerous" one with similar observation.

Therefore, a second sensor was added to detect the flow rate of the prospective hex that the scout will occupy, normalized by the highest flow rate in any single hex on the map. Since forest hexes, where the enemy is unlikely to travel, have a higher movement cost than other terrain types, the vapor flow rate through them is lower, giving a distinct signature that the network may use in its evolution process.

During later experiments, the behavior of finding concealed terrain was implemented directly during scenario configuration which forced scouts to only consider concealed positions. Therefore, the dual-sensor agent evaluated in Section 5.4 uses its second sensor detect the highest single-hex flow rate within the field of view, normalized by the highest single-hex flow rate on the map. The rationale behind this approach is that high flow rates indicate chokepoints the enemy through which the enemy is likely to travel, and thus, it may be more helpful to observe a single chokepoint than to observe a

large area through which the enemy is unlikely to go, but summed together would yield a larger flow rate than a chokepoint.

4.2.3 Directional Sensor Models

This organism has seven sensors, with the first six summing flow rates in 60 degree fields of view radiating out from the agent and the seventh sensor indicating either the flow rate of the selected hex or the highest flow rate in any hex observed, as described above. Since the simulator does not model heterogeneous sensors (i.e. being able to observe farther in one direction than the others) and scouts can only be destroyed when the enemy occupies their hex (i.e. there is no advantage to orienting towards the enemy to present a smaller silhouette or greater protection from frontal armor), there is not any obvious evolutionary advantage to more refined sensing. However, it may provide greater flexibility or lead to a more effective placement strategy than simpler networks can provide, so the directional sensor model is evaluated in the test program.

4.2.4 Comparison Baselines

Since team performance varies considerably based on the scenario map, environment characteristics, and number of units, the specific team score for a given iteration is not useful for comparison. Therefore, the performance of these agents will be compared with the performance of a baseline agent for the same map and conditions.

The first type of baseline agent will randomly choose a hex in the reconnaissance zone, representing the lowest possible hurdle to overcome. The second method is slightly more complex. Fifty random hexes in the reconnaissance zone are sampled and the scout is placed in the one that can see the largest number of hexes. Initial tests revealed that

this led to the agents clumping together in one spot leading to very poor coverage of the map. Therefore, the shared spotting fitness approach described in Section 4.3.1 was incorporated with the maximum hex view approach by discounting the value of hexes that are already observed by other units. This modified maximum hex view approach aims to observe the largest number of hexes with a bias towards those that are not already covered.

4.3 Fitness Functions

The performance of an individual organism in an iteration is evaluated according to a fitness function. At the end of each epoch, the organisms with the highest fitness are retained and mutated while those with the lowest fitness are discarded. This thesis considers individual fitness as well as bonuses to individual fitness based on the team's overall performance.

4.3.1 Simple Greedy vs. Shared Spotting

The simplest approach to fitness is a greedy approach which gives a point for every enemy unit that a scout can observe. This process repeats for every tick and the final score is normalized by the total number of possible spots (i.e. *number of enemy units* * *time spent in the reconnaissance and enemy zones*). Even though this results in very low fitness results on the scale of [0, 1], organism selection for mutation and reproduction is unaffected because organisms are chosen based on its fitness relative to that of other organisms in the population.

The second approach to fitness is shared spotting, which discounts the observed value of hexes which other scouts can observe. Since the goal is to observe enemy units

as they cross the map, there should be some incentive for scouts to disperse. Therefore, under shared spotting, the flow rate of each hex is divided by the number of friendly scouts that can see that hex. This divisor can be modified as needed to scale how strictly the user wants to penalize overlapping fields of vision.

4.3.2 Shared Team Fitness

Since the objective is for the team as a whole to perform well in spotting enemy units, another approach to fitness is to add the team's normalized performance to the unit's fitness. Since this bonus fitness is only awarded to the agents that participated in a given iteration, over time this boost should favor the evolution of agents that work together better.

4.4 Deployment Restrictions

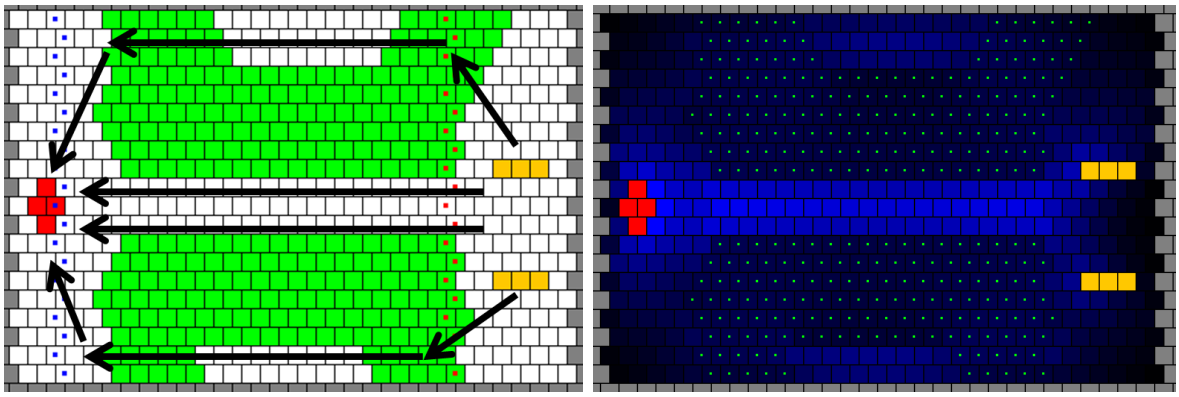
An additional parameter for experimental setup restricts scout agents to choosing a location that offers concealment (high grass or trees). This approach reflects the military doctrine that "[scouts] may need to pass up a position with favorable observation capability but with no cover and concealment in favor of a position that affords better survivability."[3] Separate baseline functions are evaluated for forced concealment and open deployment, and the performance of the scout agents is only compared with the baseline that uses the same deployment restrictions.

4.5 Scenarios

Two training scenarios and three testing scenarios were developed for this thesis. Each scenario is described below to include the reason it was developed and two maps to

help visualize the scenario. The first map indicates the terrain, possible enemy routes, and the friendly and enemy zones. The second is an equilibrium flow rate map that indicates what an organism's sensors can detect.

4.5.1 Calibration Scenario



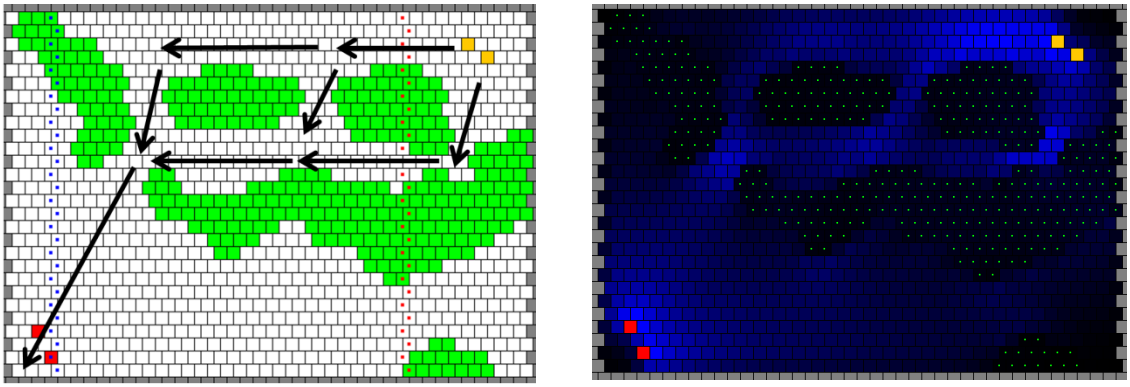
(a) (b)

Figure 4.1: Calibration scenario with possible enemy routes (a) and the vapor flow rate map (b). The large forest in the center of the map limits visibility to the central corridor and two clearings so evolved scout agents should be concentrated in or alongside those areas.

The calibration scenario (Figure 4.1) was the basis scenario used for early testing of BBMS. Each iteration had ten scouts with a maximum visibility of ten hexes. There are three possible enemy routes on the map: the central corridor and clearings in the forest along the north and south edges of the map, with the remainder of the reconnaissance zone consisting of trees. This scenario was used to determine if the scout agents were evolving properly since the only positions where agents could score fitness was either in, or alongside the central corridor or clearings. In addition to this basic validation, the calibration scenario was also used to evaluate the impact of changing the starting JNEAT parameters and network formation variables (e.g. starting number of nodes and connectivity and mutation rate). Finally, due to the limited and narrow

visibility corridors, the average death rate for scout agents is very high in this scenario. Therefore, changing the death penalty factor and restricting scout deployment would have the greatest impact to team performance on this map.

4.5.2 Training Scenario

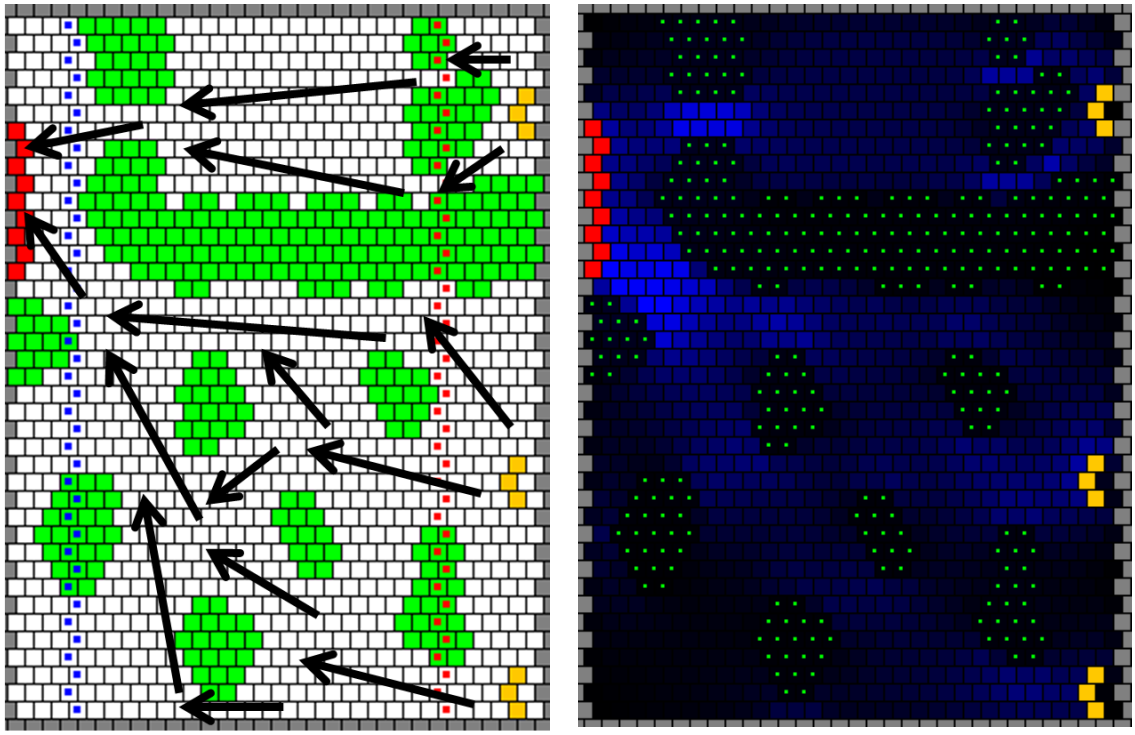


(a) (b)

Figure 4.2: Training scenario map with possible enemy routes (a) and the vapor flow rate map (b). Despite the large open area in the south central and southeast area of the map, the terrain and the enemy starting location causes enemy routes to only cover the northern and western portions of the map. Therefore, evolved agents should be concentrated in those areas of the map.

This training scenario was designed to be larger than the calibration scenario and with more diverse terrain. The enemy starting position in the northeast corner of the map and the forest running east-west along the center of the map restricts enemy movement to roughly half of the total map area. In order to make the placement of scouts more critical than in the calibration scenario, each iteration had only five scouts and maximum visibility was reduced to five hexes. The layout of the terrain and corresponding impact on the vapor flow rate should evolve organisms that concentrate in the northern half of the map.

4.5.3 Testing Scenarios



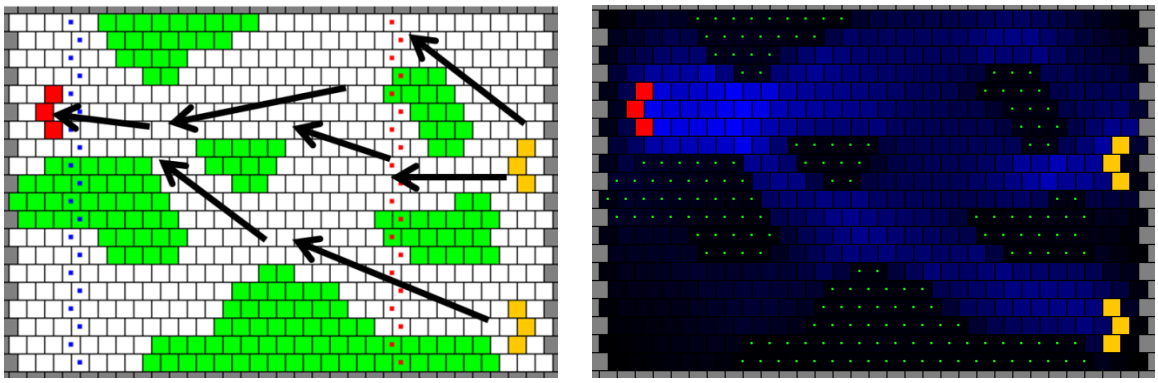
(a) (b)

Figure 4.3: Scenario 41 map with possible enemy routes (a) and the vapor flow rate map (b). This scenario was designed with a large number of corridors in the eastern half of the map to funnel into two corridors in the west.

Three maps were selected for testing, intended to cover a variety of environments to see how well the agents can generalize to different conditions. All testing scenarios have five scout agents and a maximum visibility of five hexes.

The Scenario 41 map (Figure 4.3) depicts enemy forces advancing over a very broad front. Nine enemy units are grouped by threes in the east and converge on the objective in the northwest. Enemy routes on any given iteration will cover a large portion

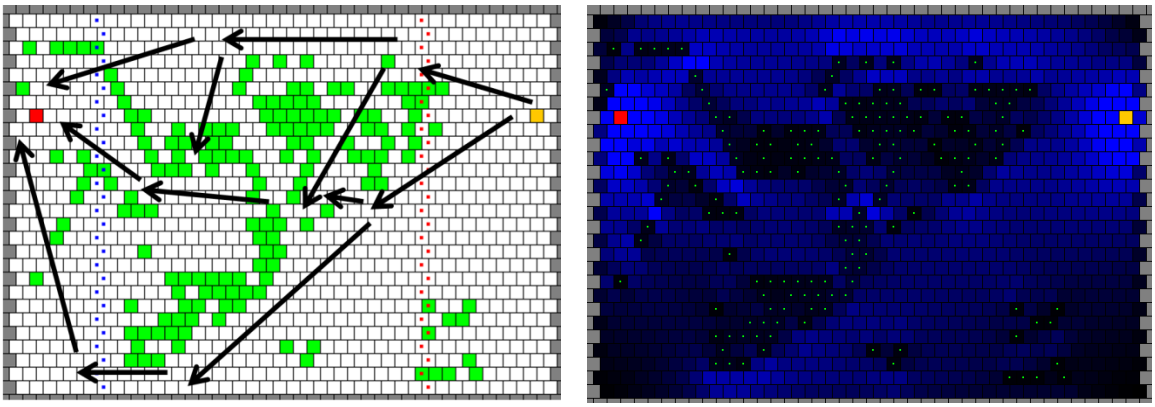
of the map and funnel through two chokepoints near the objective. This represents a target-rich environment in which there aren't enough scouts to cover the entire sector.



(a) (b)

Figure 4.4: Durango Valley map with possible enemy routes (b) and vapor flow map (b). This map uses terrain from a U.S. Army Tactical Decision Game and represents a situation in which there are a sufficient number of scouts to cover the reconnaissance zone.

The "Durango Valley" map (Figure 4.4) is taken from a US Army Tactical Decision Game published in *Armor* magazine in 1997 [15]; the only difference is that in this thesis, the hills from the Tactical Decision Game have been replaced by trees. This map was included because it represents a map developed by a third party for a military exercise. It contains six enemy units grouped by threes and due to the smaller scale compared to Scenario 41 (Figure 4.3), this represents a situation in which there are sufficient assets to screen the entire sector against a similarly sized enemy force.



(a) (b)

Figure 4.5: Steel Panthers map with possible enemy routes (a) and vapor flow map (b). This map was randomly generated from the wargame "Steel Panthers." This map is characterized by many short, narrow passages through the scattered forest and represents a situation of a small enemy unit trying to infiltrate through the screen line.

The "Steel Panthers" map was generated by the computer game "Steel Panthers: Main Battle Tank."[25] This game simulates turn-based military battles in the post-World War II era and includes a random map generator. Since BBMS does not have automatic world generation features outside of a random distributions, a random map from this computer game was converted to BBMS by replacing the buildings in the Steel Panthers map with trees (since buildings block line-of-sight and impede movement in a similar fashion to forested areas). This scenario involves a single enemy unit which represents an enemy scout vehicle trying to infiltrate through the reconnaissance zone. The scattered forest in the center of the map offers numerous concealed routes with short lines-of-sight and two wider corridors present along the northern edge of the map and angling southwest from the starting position.

4.6 CONCLUSION

The experiments in this thesis are designed to cover a wide range of parameters from sensor configurations and fitness types to the scenarios themselves. Each test scenario was chosen to reflect a different situation and terrain ranging from overstretched scouts trying to screen a sector with inadequate resources to trying to locate a lone enemy scout attempting to infiltrate the area. The variety of scenarios and parameters will provide insight into the performance of evolved scout agents in general and will set the conditions for further research in the future.

Chapter 5: Results

Each of the organism types and fitness functions in Sections 4.2, 4.3, and 4.4 were trained on the Training Scenario map and evaluated on the three testing scenarios described in Section 4.5. Their fitness during training is analyzed to determine how many epochs it takes for the networks to burn in and the impact on overall team performance. The analysis then shifts to the performance of the network on the testing scenarios and compares it to the random and maximum hex view baselines in Section 4.2.4.

5.1 Training and Testing Parameters

All neural networks were trained on the Training Scenario map with a population size of 50 with five agents per iteration and a global visibility range of five hexes. Each network was evaluated ten times after which the lower half in terms of fitness were discarded and replaced by mutated versions of the survivors. The decision to average fitness over ten iterations came from an analysis of its impact on team performance and the standard deviation of the results. This analysis, shown in Figure 5.1, reveals that averaging fitness over more than ten iterations has no impact on team performance and only a marginal decrease in the standard deviation, whereas fewer iterations increases the standard deviation and reduces performance.

The starting network topology had five hidden nodes and an initial link creation probability of 10%. In each epoch, there was a 2.5% chance for adding a link, adding a node, enabling or disabling a gene, and modifying a link weight. Each network evaluated a prospective hex according to its sensor type and fitness function, and decided to stay if

the network output >= 0.75. The training process stops after 50 epochs have been completed.

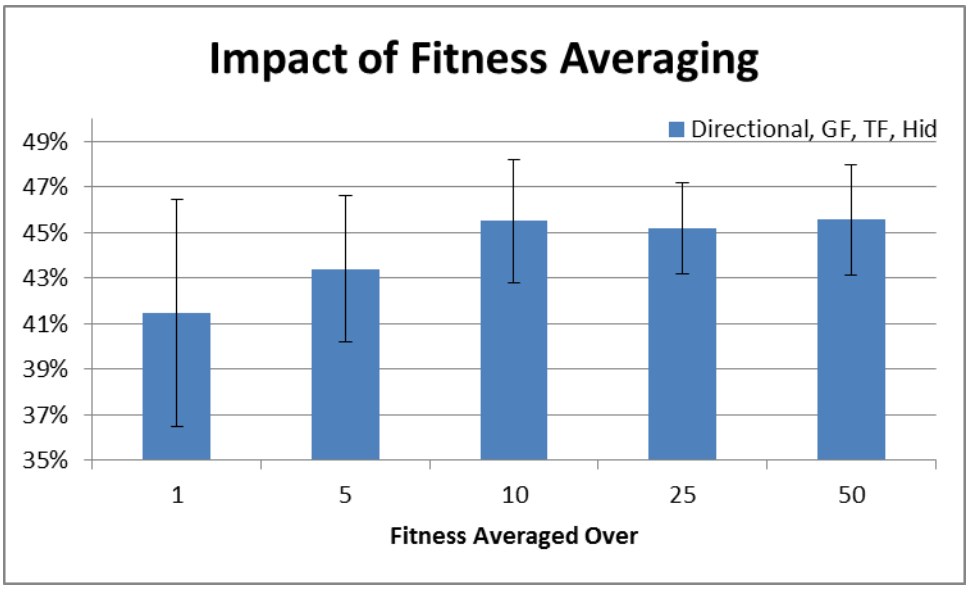


Figure 5.1 Impact of averaging fitness over several iterations per epoch. Averaging over fewer than ten iterations results in lower performance and increased standard deviation. In contrast, averaging over more than ten iterations does not improve performance or significantly reduce standard deviation.

Testing is done on the Scenario 41, Durango Valley, and Steel Panthers maps, with every fifth epoch being evaluated. Five random organisms are drawn from the population at the start of each iteration and the population is tested on each COA ten times with the team performance averaged for each COA.

5.2 Baseline Agent Performance

The random placement and maximum hex view comparison methods were evaluated on the same test suite as the standard neural networks, except they were tested on every epoch instead of every fifth one in order to have more data for averaging the

baseline results. There were two sets of base cases: One forced the scout agents to select concealed positions and the other allowed the scouts to be placed anywhere in the recon zone.

	Performance / Standard Deviation		Death Rate	
Scenario	**Concealed**	**Open**	**Concealed**	**Open**
Scenario 41	8.545% / 1.65%	15.055% / 1.49%	0.14%	17.38%
Durango Valley	17.814% / 3.47%	26.265% / 3.26%	0%	17.32%
Steel Panthers	13.275% / 4.34%	13.159% / 1.16%	0.33%	3.22%

Table 5.1: Random Placement team performance.

	Performance / Standard Deviation		Death Rate	
Scenario	**Concealed**	**Open**	**Concealed**	**Open**
Scenario 41	25.031% / 8.60%	25.278% / 4.88%	0.08%	38.56%
Durango Valley	57.770% / 7.37%	40.614% / 6.54%	0%	35.52%
Steel Panthers	27.170% / 4.18%	24.184% / 7.05%	0.13%	3.95%

Table 5.2: Maximum Visible Hex team performance.

The performance of the random placement baseline in Table 5.1 represents the absolute lowest performance hurdle for evolved scout agents. If the evolved scout agents are unable to outperform the randomly placed scouts, that would indicate that the gas diffusion method has no relation to the movement routes of the enemy forces. Forcing the scouts to find concealment significantly reduces the performance of randomly placed agents because a greater portion of available deployment areas are surrounded by trees. The only hexes that would provide greater view range are on the perimeter of the forests. The team performance of the random baseline improves considerably when the concealment restriction is removed but results in a moderately high death rate. The Steel

Panthers scenario has a much lower death rate because it only has one enemy unit, significantly reducing the chance that any given scout will lie directly in the enemy's path.

The maximum hex view performance in Table 5.2 reveals that concealed deployment offers similar or better performance than open deployment. This is because the maximum hex view heuristic avoids deployment in the center of the forest and places units on the edges of forests. This placement offers good vision to at least the front half of the scout and drops the death rate down to near-zero. In contrast, open placement will invariably deploy away from the forest and will tend to find locations in the middle of an open area so it can see the most hexes. The center of large maneuver corridors are the preferred routes for enemy vehicles and therefore the death rate of maximum hex view agents in the open is double that of the random agents except on the Steel Panthers map.

Subsequent figures will evaluate the evolved scout agents against these two baselines and differentiate between open and concealed scout placement.

5.3 SINGLE-SENSOR AGENTS

Single-sensor agents take the sum of vapor flow rates within their field of view and normalize it by the highest vapor flow sum of any position on the map. Figure 5.1 shows a typical training profile for a single-sensor scout agent with the training performance of each type of single-sensor agent shown in Table 5.3. In the training profile, there is a significant improvement in team performance after the first epoch representing burn-in, but there is no significant change to performance in the following epochs. The shared spotting fitness function significantly increases the overall performance in the training scenario by about five percentage points (see Table 5.3) and

forcing the scouts to find concealment drops the 9.8% death rate to 0% as well as improving overall team performance by approximately 12 percentage points.

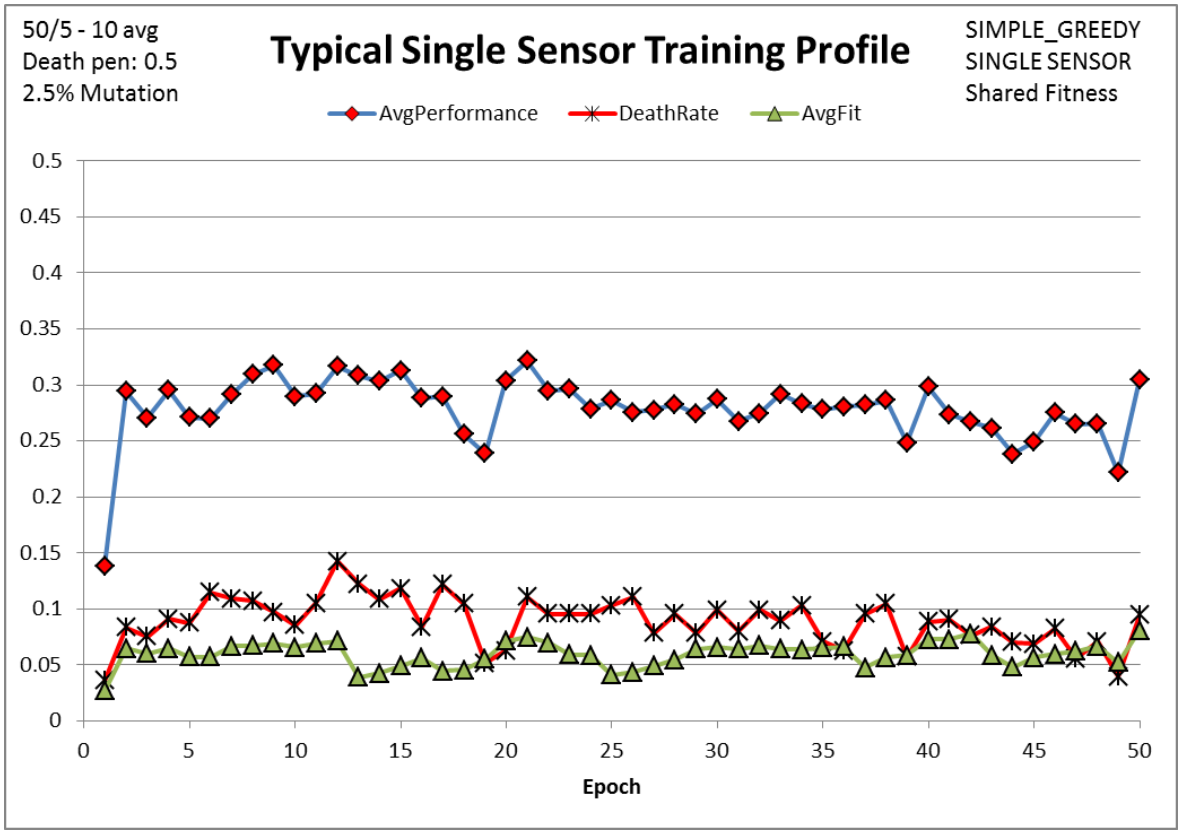


Figure 5.2: Typical single-sensor training profile on Scenario 51. All eight configurations show the same single epoch burn-in followed by no further improvement.

Force Conceal	Share Fit	Performance/STDEV (Simple Greedy)	Performance/STDEV (Shared Spotting)
No	No	26.084% / 1.93%	32.366% / 3.14%
No	Yes	28.174% / 2.13%	32.892% / 2.53%
Yes	No	40.809% / 3.04%	45.174% / 1.99%
Yes	Yes	39.331% / 3.01%	44.911% / 3.44%

Table 5.3: Single-sensor training performance. Performance improves significantly under the shared spotting fitness function and when forcing concealed deployment.

In the test cases, all types of single-sensor agents outperformed the random baseline following burn-in. Figure 5.2 shows the team performance averaged over the three testing scenarios for the eight configurations of single sensor agents. The shared spotting fitness function overall performed better than the greedy fitness, which after 45 epochs had an overall team fitness 5.3 percentage points higher than the random baseline in unrestricted deployment. However, even this top-performing configuration performed consistently worse than the maximum hex view baseline by 5.6 percentage points. If deployment is restricted to tree and tall grass hexes, relative performance drops further and falls 2-3 percentage points below random deployment and a much larger 19-21 percentage points below maximum hex view.

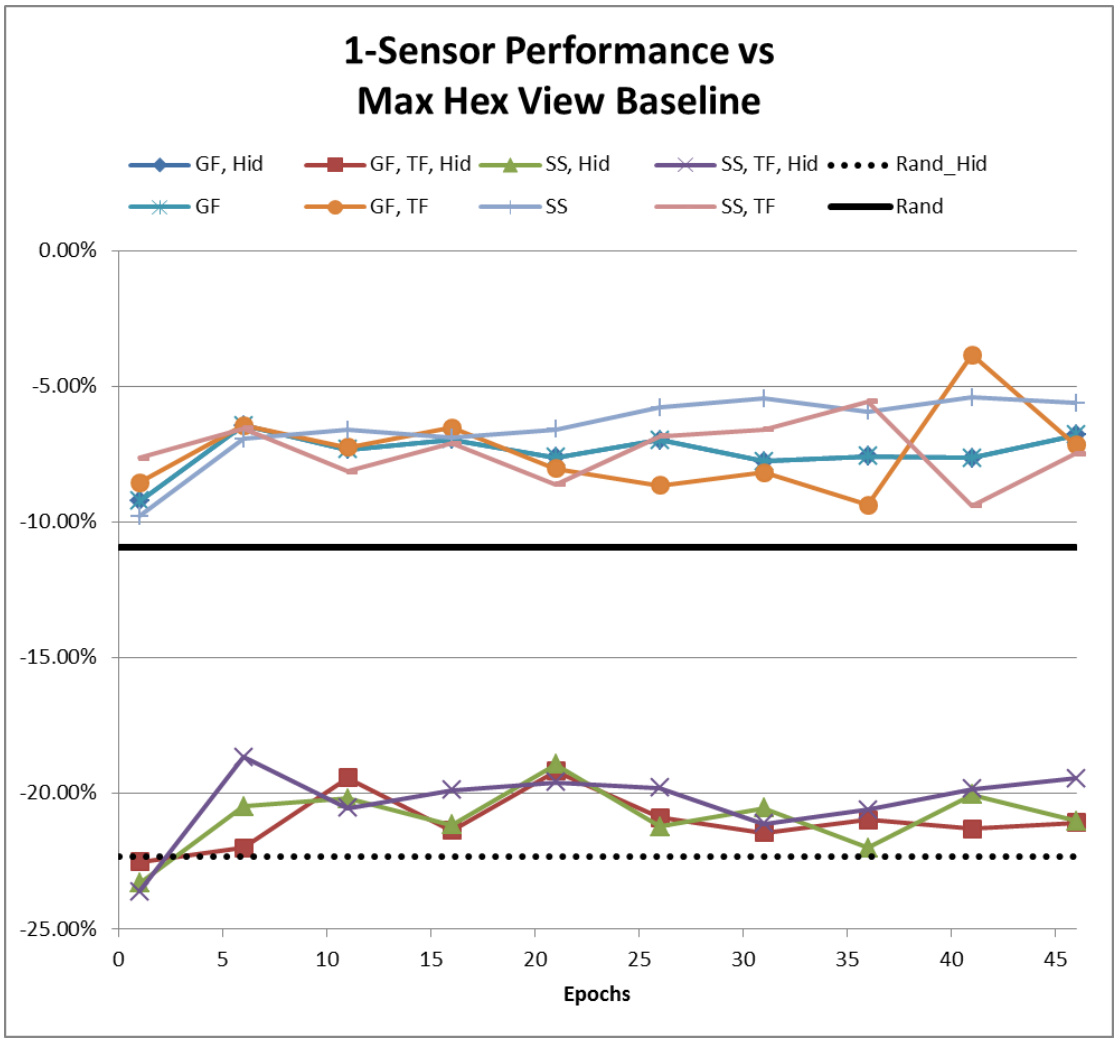


Figure 5.3: Comparison of overall-single sensor agent performance versus random and maximum hex view baseline methods. Legend key: GF = greedy fitness, SS = shared spotting, TF = team fitness sharing, Hid = force deployment to concealed hexes, Rand is the average performance of the random baseline. After the initial epoch, all single-sensor scout agents outperform their respective random baseline indicating that the gas vapor model is indicative of enemy routes. However, there is no single-sensor configuration that consistently outperforms the maximum hex view baseline with concealed agents performing over twenty percentage points below that baseline.

A closer look at the results revealed that the performance of these agents varies considerably depending on which map and course of action they face (see Table 5.4).

Specifically, these networks all perform well in the Steel Panthers scenario, particularly the COA in which the enemy unit sneaks through the narrow paths in the center of the map. This is a situation in which a narrow corridor has a sufficiently high flow rate to outweigh the limited number of hexes visible in there. In contrast, units on the other maps pass through wider corridors making it much more likely for a max hex view agent to spot it.

GF, Hid	Scenario 41					Durango Valley				Steel Panthers			
EPOCH	Balance	M-Cent	M-Nor	M-Sout	Scout	Edges	North1	North2	South	S-Cent	S-Nor	S-South	S-Flank
1	-10.2%	-24.9%	-3.5%	-21.5%	-19.6%	-32.6%	-49.4%	-43.0%	-47.4%	-17.8%	-5.7%	-9.1%	-24.4%
6	-11.8%	-23.8%	-3.5%	-20.6%	-17.0%	-36.8%	-54.2%	-38.6%	-39.0%	-4.6%	7.1%	-7.7%	-16.7%
11	-10.6%	-26.2%	-4.0%	-24.1%	-20.4%	-33.9%	-48.2%	-37.1%	-33.5%	-9.6%	-2.1%	-3.2%	-16.2%
16	-11.2%	-28.9%	-6.9%	-20.7%	-19.3%	-40.5%	-46.9%	-40.0%	-37.6%	-17.6%	-0.5%	-8.9%	-18.9%
21	-12.4%	-27.9%	-5.1%	-21.1%	-18.4%	-27.3%	-44.5%	-47.9%	-47.5%	-12.0%	2.8%	-7.1%	-6.5%
26	-11.0%	-25.9%	-3.4%	-22.3%	-21.0%	-32.1%	-47.0%	-39.5%	-38.2%	-3.7%	6.1%	2.0%	-2.7%
31	-9.4%	-27.4%	-1.3%	-23.5%	-19.2%	-34.3%	-51.0%	-41.2%	-34.9%	-6.8%	5.9%	0.4%	-4.9%
36	-8.5%	-21.5%	-2.5%	-18.9%	-20.7%	-38.2%	-51.2%	-45.2%	-37.4%	-5.3%	-0.1%	-5.0%	-9.4%
41	-11.7%	-18.7%	-5.4%	-23.4%	-22.7%	-39.2%	-43.7%	-43.8%	-32.8%	-8.1%	5.3%	3.8%	-15.8%
46	-11.1%	-27.1%	-0.5%	-21.6%	-17.6%	-32.8%	-47.9%	-41.1%	-41.9%	-10.0%	-1.3%	0.4%	-16.5%
AVG 6+	**-10.9%**	**-25.3%**	**-3.6%**	**-21.8%**	**-19.6%**	**-35.0%**	**-48.3%**	**-41.6%**	**-38.1%**	**-8.6%**	**2.6%**	**-2.8%**	**-12.0%**

Table 5.4: Relative percentage point difference in team performance for a GF/Hid agent compared to the maximum hex view baseline. The bottom is the average performance excluding epoch 1 due to burn-in. Note the widely varying performance depending on the map and course of action.

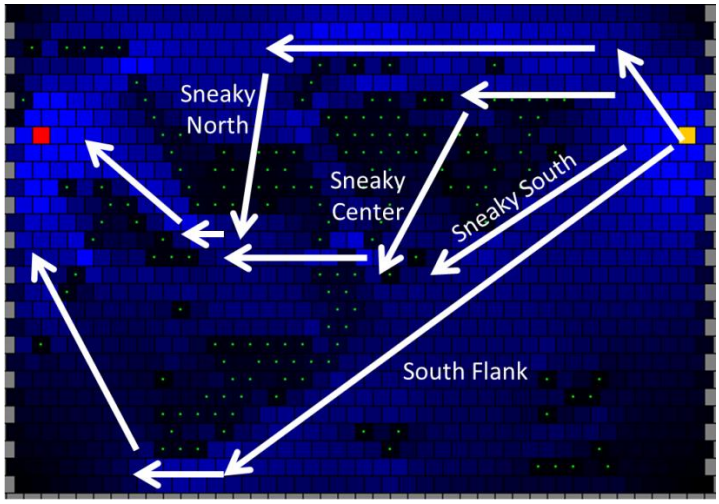


Figure 5.4: Labeled COAs for Steel Panthers test with vapor flow overlay. Note that the worst performance vis a vis max hex view is on the "South Flank" COA because the large open space on the south flank is attractive to that heuristic whereas the "Sneaky Center" COA is not.

5.4 DUAL-SENSORS

The dual-sensor networks evaluated here have their second sensor attuned to the highest flow rate within the agent's field of view. Figure 5.4 shows the typical training profile for the dual-sensor networks over all eight configuration types. The burn-in period for dual-sensor networks is generally longer than for single-sensor networks, though still fairly short at one to three epochs. As with the single-sensor agents, forcing the agents to find a concealed spot drops the death rate from 7-20% for this class of agent down to 0%.

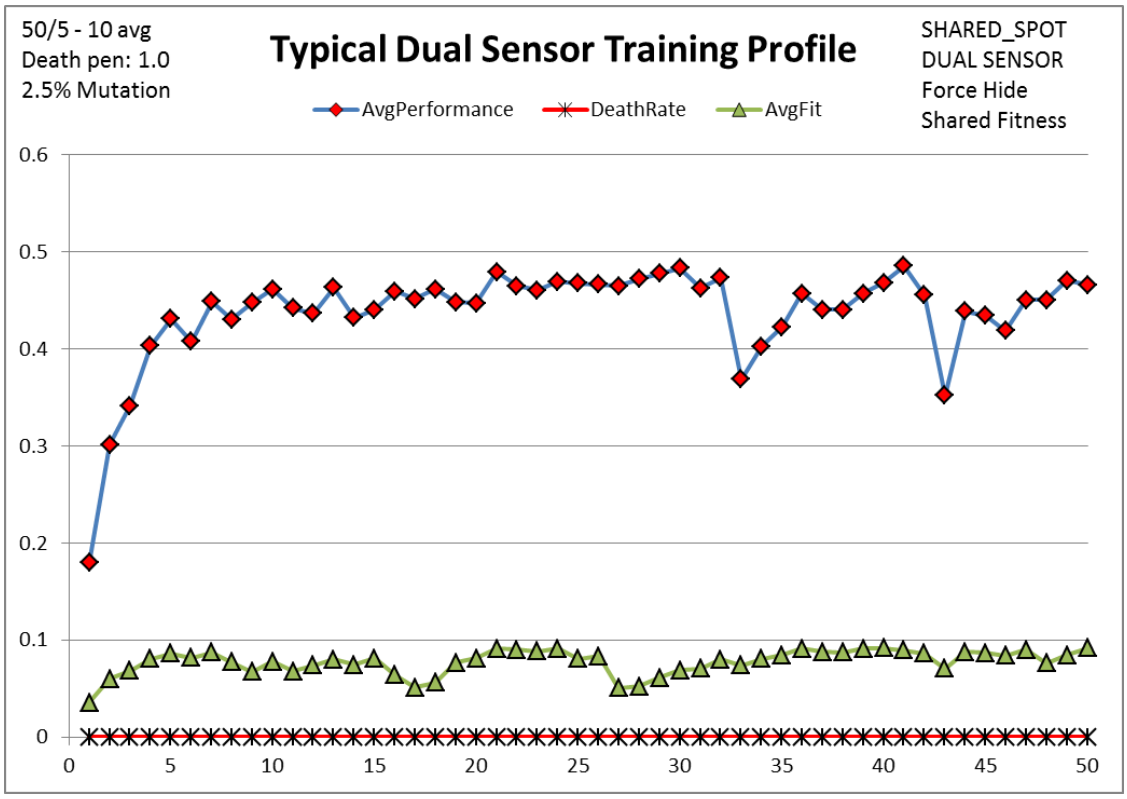


Figure 5.5: Typical dual-sensor training profile from Scenario 51. Scenario parameters are at the top. Note the longer burn-in compared to single-sensor. Also, since this configuration forces deployment in concealed hexes, the death rate is extremely low.

Force Conceal	Share Fit	Performance/STDEV (Simple Greedy)	Performance/STDEV (Shared Spotting)
No	No	31.136% / 3.16%	37.755% / 3.68%
No	Yes	32.125% / 1.81%	36.555% / 3.63%
Yes	No	37.178% / 3.87%	44.794% / 3.23%
Yes	Yes	38.015% / 3.54%	44.765% / 3.70%

Table 5.5: Dual-sensor training performance. Performance compared to single-sensor training performance is similar for forced concealment, but four to five percentage points better for unrestricted deployment.

Performance on the training scenario shows that the dual-sensor approach yields comparable performance to single-sensor agents when forced to find a concealed location. However, the second sensor improves performance by four to five percentage points on unrestricted placement.

The dual-sensor agents across the board perform better than single-sensor agents by about five percentage points (see Figure 5.4). As before, dual-sensor agents consistently outperform random placement with unrestricted deployment seeing the best relative gains. Furthermore, the unrestricted shared spotting fitness function reaches one percentage point shy of the max hex view baseline.

SS, TF, Hid	Scenario 41					Durango Valley				Steel Panthers			
EPOCH	Balance	M-Cent	M-Nor	M-Sout	Scout	Edges	North1	North2	South	S-Cent	S-Nor	S-South	S-Flank
1	-10.5%	-27.3%	-4.5%	-22.4%	-22.2%	-34.8%	-46.0%	-39.4%	-39.4%	-11.3%	-5.9%	-8.2%	-20.9%
6	-9.9%	-25.9%	-5.0%	-16.0%	-17.0%	-37.9%	-40.5%	-34.5%	-32.9%	-3.5%	14.6%	-3.2%	-11.8%
11	-12.3%	-20.3%	-2.7%	-19.0%	-18.4%	-31.2%	-48.4%	-40.0%	-37.5%	-8.1%	3.8%	1.1%	-12.7%
16	-15.1%	-27.5%	-3.4%	-23.8%	-21.1%	-32.6%	-50.0%	-41.9%	-37.9%	-12.4%	6.3%	-2.1%	-12.9%
21	-9.9%	-20.4%	-7.2%	-22.3%	-20.0%	-29.9%	-45.4%	-33.8%	-33.4%	-7.0%	6.7%	-3.9%	-14.0%
26	-9.5%	-25.6%	-2.7%	-16.7%	-20.1%	-33.4%	-49.9%	-39.0%	-43.9%	-8.7%	3.8%	-4.3%	-14.2%
31	-10.4%	-28.9%	-6.4%	-18.5%	-17.6%	-34.7%	-57.1%	-43.7%	-39.7%	-10.5%	-7.9%	-4.1%	-16.9%
36	-10.4%	-22.6%	-6.0%	-21.1%	-18.7%	-35.7%	-52.2%	-47.0%	-29.0%	-13.3%	-0.7%	-3.4%	-14.0%
41	-12.1%	-27.5%	-7.0%	-22.3%	-21.0%	-30.5%	-49.5%	-38.6%	-48.1%	-9.8%	-6.9%	0.0%	-8.5%
46	-12.0%	-24.7%	-4.4%	-18.0%	-20.9%	-37.7%	-47.9%	-42.5%	-37.2%	-7.6%	4.7%	-5.7%	-17.8%
AVG 6+	**-11.3%**	**-24.8%**	**-5.0%**	**-19.7%**	**-19.4%**	**-33.7%**	**-49.0%**	**-40.1%**	**-37.7%**	**-9.0%**	**2.7%**	**-2.8%**	**-13.7%**

Table 5.6: Relative percentage point difference in team performance between the best performing dual-sensor agent and the max hex view comparison method. As with single-sensor agents, performance exceeds that of max hex view on the Steel Panthers S-North COA and comes close on the Steel Panthers S-South and Scenario 41 M-North.

SS, TF	Scenario 41					Durango Valley				Steel Panthers			
EPOCH	Balance	M-Cent	M-Nor	M-Sout	Scout	Edges	North1	North2	South	S-Cent	S-Nor	S-South	S-Flank
1	-10.3%	-6.1%	-7.0%	-7.8%	-14.4%	-15.1%	-19.9%	2.2%	1.8%	-0.6%	-4.3%	-15.9%	-7.6%
6	-9.6%	0.7%	-7.1%	-11.1%	-14.1%	-15.9%	-12.7%	6.3%	0.3%	3.7%	-2.4%	-8.4%	-15.6%
11	-8.8%	-2.9%	-5.2%	-11.9%	-12.4%	-7.5%	-9.1%	2.9%	-3.8%	13.0%	5.6%	-12.5%	-7.8%
16	-9.0%	-8.1%	-2.6%	-13.1%	-14.0%	-14.1%	-17.2%	-4.9%	-5.0%	2.2%	4.8%	-7.9%	-7.6%
21	-9.8%	-6.4%	-3.8%	-13.9%	-17.0%	-14.1%	-16.5%	-1.5%	-5.9%	-1.9%	-1.6%	-16.6%	-8.7%
26	-10.4%	-4.9%	-6.8%	-12.0%	-13.5%	-16.0%	-20.0%	9.7%	-3.4%	5.4%	-2.4%	-10.9%	-6.9%
31	-8.8%	0.3%	-7.0%	-11.9%	-14.9%	-11.5%	-13.0%	12.4%	-12.0%	-1.5%	5.6%	-10.0%	-0.5%
36	-5.6%	-2.7%	-5.9%	-15.2%	-15.8%	-10.9%	-10.2%	3.0%	-4.3%	-3.9%	3.8%	-9.8%	-11.2%
41	-9.4%	-4.6%	-0.3%	-14.9%	-12.8%	-12.2%	-11.3%	3.3%	-0.2%	1.8%	5.0%	-13.4%	-8.1%
46	-4.6%	5.3%	-7.1%	-9.5%	-9.8%	-10.3%	-9.7%	3.6%	3.1%	0.0%	7.7%	-10.2%	-3.6%
AVG 6+	**-8.4%**	**-2.6%**	**-5.1%**	**-12.6%**	**-13.8%**	**-12.5%**	**-13.3%**	**3.9%**	**-3.5%**	**2.1%**	**2.9%**	**-11.1%**	**-7.8%**

Table 5.7: Same type of agent as in Table 5.6, but without forcing concealment. Note that there is a separate max hex view baseline for concealed and unrestricted agents.

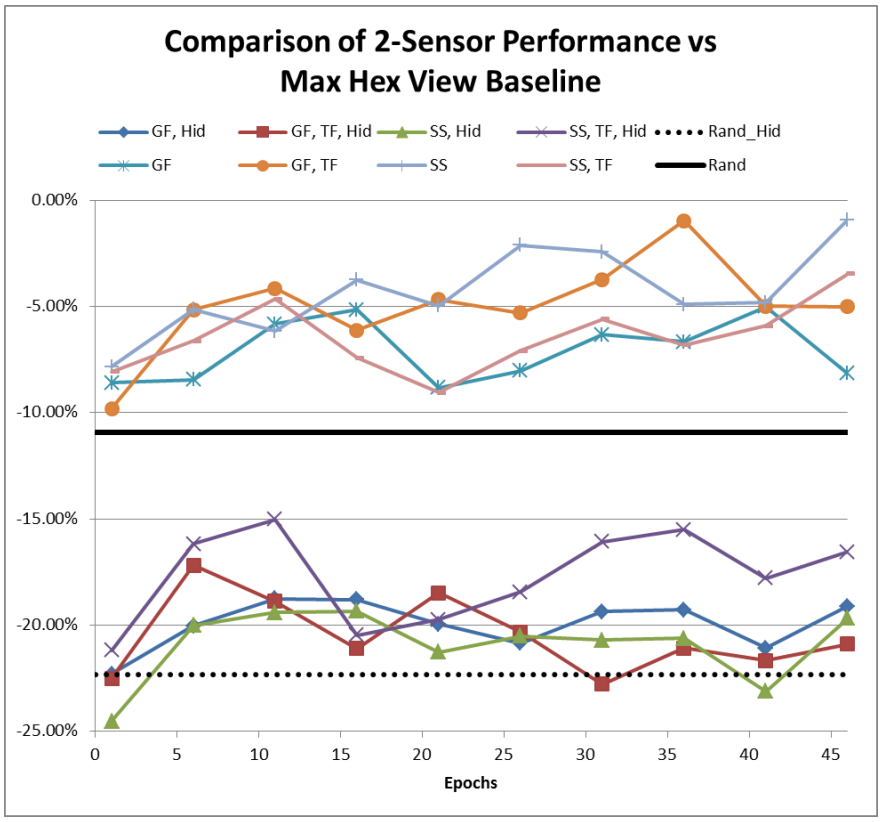


Figure 5.6: Comparison of overall sensor performance versus the random and max hex baselines.

The relative performance by COA of a hidden dual-sensor agent in Table 5.6 is very similar to that of a hidden single-sensor agent in Table 5.4, outperforming maximum hex view on the Steel Panthers S-North COA and coming close in the Steel Panthers S-South and Scenario 41 M-North COAs. Table 5.7 shows the performance by COA of an unrestricted dual-sensor agent, which has significantly better relative performance in Durango Valley, particularly COA North2 and COA South. These results show that the dual-sensor approach offers an incremental improvement to the team's performance but does not change the underlying method that the population evolves over time.

5.5 DIRECTIONAL SENSORS

The directional sensor model exhibits some unique differences in training from the single and dual sensor models. The typical training profile diverges between the shared spotting fitness function and the simple greedy function. The greedy approach yields a performance profile similar to that of the single- and dual-sensor organism with performance stabilizing within the first few generations and remaining largely unchanged for the remaining training epochs. In contrast, three of the four directional sensor implementations with shared spotting fitness (see Figure 5.6) continued to improve in small increments through the first 30-35 generations.

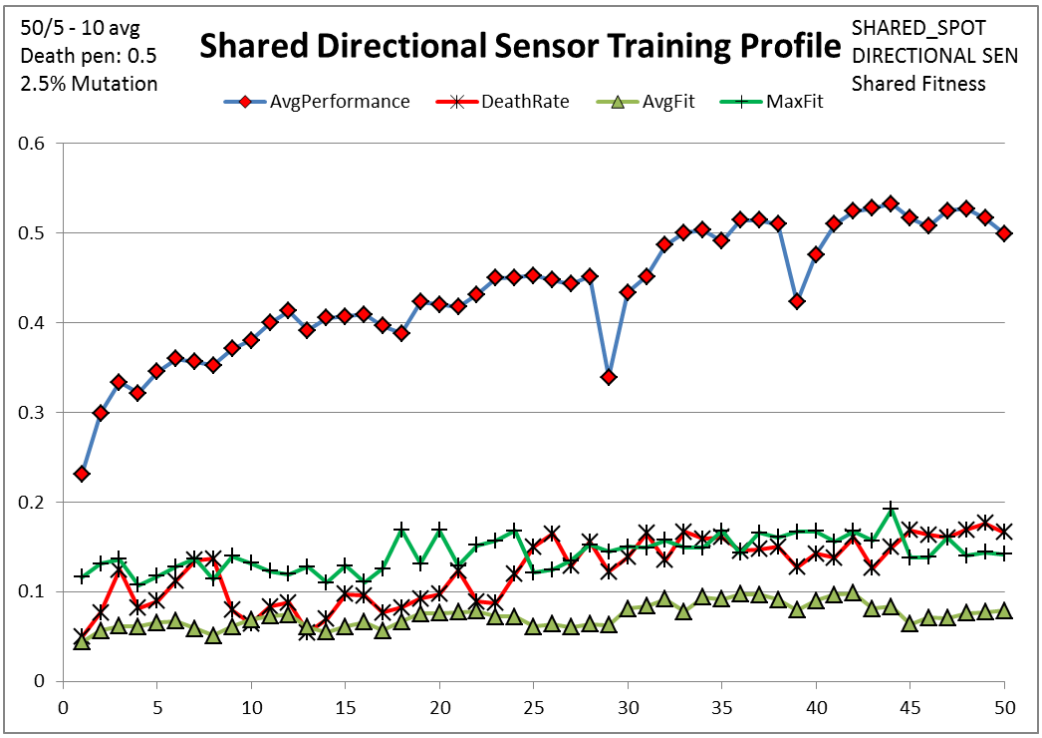


Figure 5.7: Progressive learning in a directional sensor shared spotting agent. In contrast to previous training profiles, team performance improves over the first thirty epochs before stabilizing.

The simple greedy directional agents perform roughly on par with the dual agents though the unrestricted agents have a significantly lower death rate averaging about 7.5% compared to 14.6% for dual unrestricted agents. The main performance difference was found in the shared spotting directional agents. Taking into account the long burn-in time, the average performance of the network, once it matures, is up to six percentage points higher than the best performing dual network.

Force Conceal	Share Fit	Performance/STDEV (Simple Greedy)	Performance/STDEV (Shared Spotting)
No	No	33.84% / 2.15%	47.19% / 3.61%
No	Yes	37.15% / 2.74%	49.97% / 2.49%
Yes	No	36.75% / 1.71%	50.97% / 2.88%
Yes	Yes	37.87% / 1.48%	46.74% / 1.26%

Table 5.8: Directional sensor training performance, with a burn-in of 30 generations. The shared spotting training performance of 46-50% represents a 6 to 13 percentage point improvement over training performance of dual agents.

The superior performance of the directional sensor model in training does not carry over to the test suite (see Figure 5.7). Its best performance is about six percentage points below the max hex view baseline and about four percentage points below the best dual sensor agents. When considering performance in individual COAs (Table 5.9), the dual-sensor and directional sensor populations outperform the max hex view baseline on the same courses of action, though the directional sensor population does particularly poor against the "S-Flank" scenario in "Steel Panthers" averaging 25 percentage points below the baseline, as opposed to only eight percentage points below for the dual-sensors.

SS	Scenario 41					Durango Valley				Steel Panthers			
EPOCH	Balance	M-Cent	M-Nor	M-Sout	Scout	Edges	North1	North2	South	S-Cent	S-Nor	S-South	S-Flank
1	-8.2%	-6.6%	-5.7%	-8.9%	-14.0%	-11.4%	-11.9%	6.4%	0.6%	3.7%	-4.3%	-15.7%	-17.4%
6	-11.4%	-8.3%	-7.4%	-10.8%	-19.7%	-10.0%	-15.5%	10.4%	-7.9%	3.3%	-0.2%	-5.9%	-18.3%
11	-8.2%	-9.7%	-7.1%	-12.5%	-16.4%	-17.0%	-9.8%	0.4%	-8.6%	10.4%	7.3%	-9.1%	-25.3%
16	-10.3%	-1.8%	-5.2%	-2.9%	-17.9%	-10.7%	-8.0%	15.2%	-4.6%	8.9%	9.4%	-4.1%	-29.3%
21	-10.7%	-10.0%	-8.6%	-16.4%	-13.6%	-8.5%	-13.6%	6.8%	-1.1%	5.0%	9.6%	-1.8%	-27.8%
26	-9.0%	-3.3%	-1.6%	-12.7%	-17.5%	-12.6%	-10.2%	2.0%	-10.9%	3.9%	5.4%	-6.1%	-21.6%
31	-10.0%	-0.5%	-5.1%	-13.9%	-14.4%	-9.7%	-14.0%	11.0%	-11.1%	11.3%	13.5%	-3.2%	-22.0%
36	-10.2%	1.2%	-6.1%	-11.6%	-20.9%	-10.5%	-16.0%	10.3%	-4.0%	10.8%	4.8%	-2.9%	-24.9%
41	-7.3%	4.2%	-7.7%	-9.7%	-12.9%	-10.6%	-16.5%	9.8%	-14.0%	7.0%	0.5%	0.7%	-31.8%
46	-10.3%	4.3%	-6.8%	-3.0%	-12.1%	-11.3%	-14.3%	6.9%	-14.5%	5.2%	1.7%	0.5%	-25.8%
AVG 6+	**-9.7%**	**-2.7%**	**-6.2%**	**-10.4%**	**-16.2%**	**-11.2%**	**-13.1%**	**8.1%**	**-8.5%**	**7.3%**	**5.8%**	**-3.5%**	**-25.2%**

Table 5.9

Table 5.9: Relative percentage point difference in team performance compared to max hex view baseline. This configuration outperforms maximum hex view on the same COAs as dual-sensor shared spotting performance (Table 5.7) and does particularly well on Durango Valley COA North 2 and Steel Panthers COA S-Center.

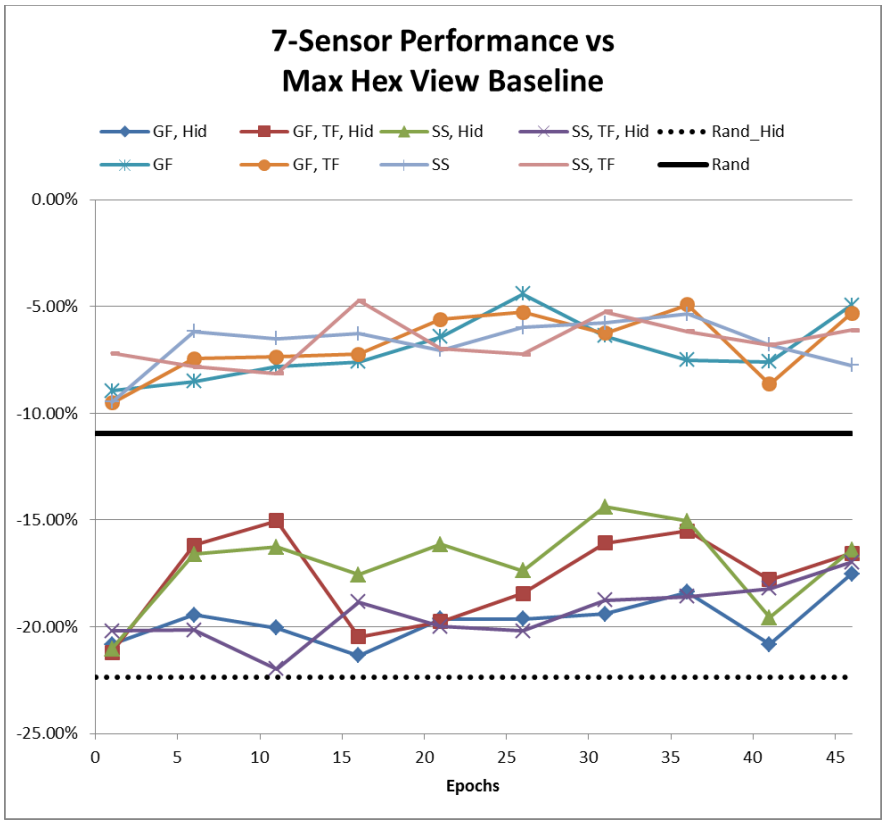


Figure 5.8: Comparison of overall sensor performance versus the random and max hex baselines. Despite significantly higher training performance, relative performance averaged over all testing COAs is similar to that of dual-sensor networks in unrestricted deployment. Restricted deployment of directional networks does not offer a higher maximum performance compared to dual-sensor networks but there is less variance between the different configurations.

5.6 Impact of Training Scenario

A final series of experiments were run to determine how much the training scenario affected testing performance. The dual- and directional sensor configurations

with shared spotting, team shared fitness, and forced concealment were selected for this evaluation because they offered the best overall performance from the tests in Section 5.3, 5.4, and 5.5. New populations with these configurations were trained on each of the four scenarios and then tested on the same four scenarios (Figure 5.9).

Surprisingly, the choice of training scenario only has a minor impact on the resulting team performance. Furthermore, there is no advantage to performance when agents are evaluated on the same scenario they were trained on. Based on the results, training on Durango valley yields slightly better performance, the training scenario and Scenario 41 yield average performance, and Steel Panthers offers slightly worse performance.

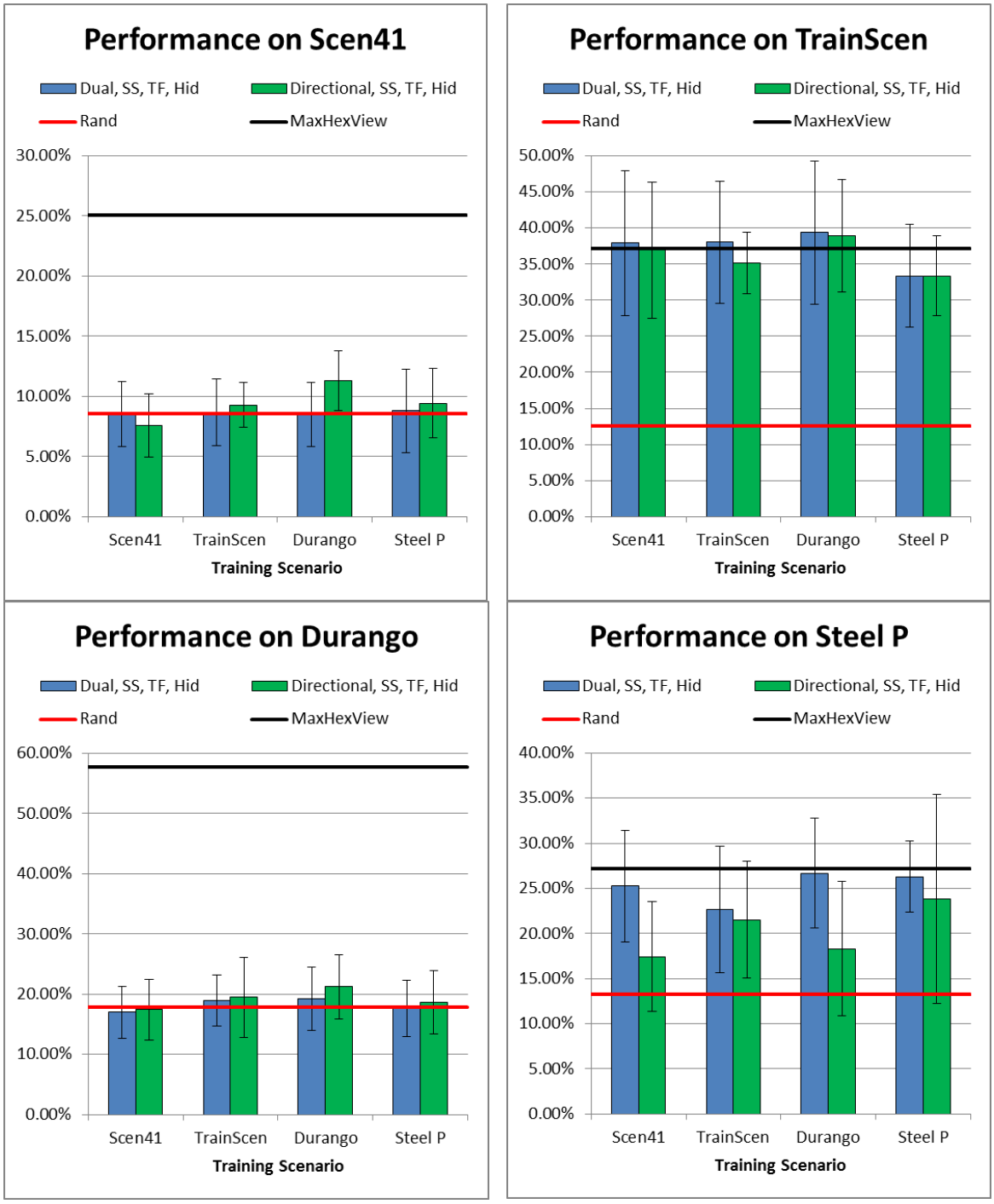


Figure 5.9: Performance by scenario compared to the training scenario. Durango Valley is the training scenario that produced the best results in testing and surprisingly there is no advantage if the evaluation and testing map are the same.

5.7 CONCLUSION

The dual- and directional sensor configurations delivered the best performance of the scout agents considered, both consistently outperforming randomly placed agents regardless of scenario. Furthermore, the shared spotting fitness function usually performed better than greedy fitness. On the other hand, adjusting individual fitness based on team performance produced inconsistent results, sometimes considerably improving the results (such as concealed dual-sensor shared-spotting in Figure 5.5) and sometimes performing worse than individual fitness (such as concealed directional-sensor shared spotting in Figure 5.7).

Chapter 6: Discussion and Future Work

6.1. Performance Relative to Baseline

As currently implemented, evolving agents to place scouts based on the gas diffusion model all show an improvement over random placement, showing that the flow rate is at least somewhat helpful in placing units. However, its inability to outperform the maximum hex view heuristic in most cases shows that more refining and testing needs to be done before these types of agents become feasible. In particular, there are certain maps and COAs in which the evolved agents consistently outperformed the maximum hex view heuristic, such as situations in which the enemy forces are only advancing through a portion of the map instead of a broad-front approach.

For example, on the scenario "Durango Valley" the unrestricted agents performed considerably better against the maximum hex view baseline in COA North 2 versus COA North 1. A look at the enemy unit paths for these COAs (Figure 6.1) shows the main difference is that the southern valley is unused in COA North 2. Doing further testing on maps and COAs categorized by the enemy unit coverage (the portion of the map that enemy units see and can be seen during their routes) will help determine if that is the main contributing factor to outperforming maximum hex view.

Another way to improve the performance of the scout agents is to evolve them using several maps with different types of terrain on them. Due to the short burn-in time of all but the shared spotting directional organisms, this would not significantly extend the length of the training period compared to what was tested in this thesis. Care would need to be taken to ensure that the networks are not overtrained on any individual map.

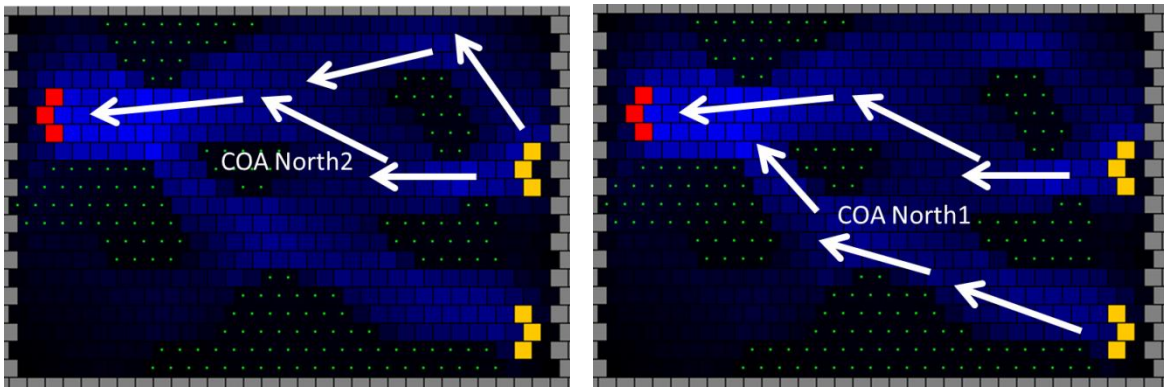


Figure 6.1: Comparison of COA North 1 and COA North 2 on the Durango Valley map. Unrestricted scout agents outperformed the maximum hex view baseline in COA North 2 yet performed poorly against the same baseline in COA North 1, possibly because COA North 1 covers a greater portion of the map.

6.2 Comparison of Organism Types

Dual- and directional networks consistently matched or outperformed single-sensor networks in these experiments. In the domain of unrestricted deployment, the best dual-sensor network configuration had considerably better performance (about four percentage points) than the best directional sensor network configuration, but the results also had greater variance from epoch to epoch than the directional networks. When deployment is restricted to concealed hexes, directional agents of all types performed three to four percentage points better than dual agents.

Of the fitness functions evaluated, shared spotting generally outperformed simple greedy, but there are some instances such as the greedy/team fitness dual-sensor configuration and the restricted directional sensor configurations that the greedy function performance matched or exceeded shared spotting. Sharing team fitness generally led to marginal, if any, improvements in team performance. There is one notable exception: The hidden dual sensor with shared spotting outperformed its non-team sharing counterpart by five percentage points.

6.3 USEFULNESS OF THE GAS DIFFUSION METHOD

The gas diffusion method accurately identifies chokepoints along the most likely enemy routes between their starting point and objective. Furthermore, improvements to Burgess' implementation [1] resulted in an eightfold improvement in the time required to find an equilibrium state on a new map. Faster performance makes it easier to employ the gas diffusion method and update the vapor map, such as after discovering enemy units in an unexpected location.

The vapor flow rates at equilibrium state are good indicators of chokepoints in the terrain, but also are high in the area immediately surrounding the source and sink. If a scout can observe these high flow rates around the source and sink, it may give undue weight to observing the source and sink compared to chokepoints in the terrain. If the source and sink represent concrete enemy starting and ending locations, then this is a good thing because observing the source and sink will guarantee detection of the enemy. On the other hand, if the source and sink represent best guesses as to the enemy locations and objectives, the high flow rates could lure scout agents away from observing chokepoints in the center of the map.

To prevent the flow rates around the source and sink from having undue influence, vapor flow rates seen by the scout agents should be discounted around those areas by a factor that represents the certainty that it represents the physical starting and ending positions of enemy forces. The vapor sink in particular should be considered for modified flow rates because the enemy does not always have to physically occupy an objective in order to accomplish their mission (e.g. if the enemy forces are trying to

isolate the objective, they can accomplish this by securing the routes to and from the objective while avoiding the objective itself).

6.4 Sequential Deployment of Sensors

Currently, the deployment of all scout agents at the start of the scenario represents either fixed sensors emplaced or left behind by other forces, or units that have been in the area for a long time and have developed fixed observation points. If we change deployment so that additional scouts become available throughout the evaluation, and update the vapor density map as enemy units are detected (e.g. by having a new unit represent a mobile vapor source), a smaller number of scouts would be needed to cover the same area and react to enemy movement. Some scouts would occupy a screen line forward to initially detect the enemy, allowing subsequent units to be placed based on what they can observe.

The issue of non-real time vapor equilibrium can be mitigated by continuing to update the vapor map during the scenario. This would cause the vapor map to gradually change in response to the spotted enemy units, with several vapor map updates every tick.

6.5 Mobile Sensors and Counter Detection

The step beyond sequential sensor deployment would be to enable the scout agents themselves to move, e.g. in response to an updating vapor map described above. Another option is to add more outputs to the network similar to the roving eye evolved for "Go", so that scouts can move and would be able to trail enemy units. Adding more realistic rules for enemy counterdetection and destroying scouts from a distance and

allowing scouts to move would better simulate soldiers on the ground as opposed to the current agent behavior which is more akin to immobile sensors than a manned position.

6.6 Adversarial Testing

Currently, enemy courses of action are static and have to be hand-coded by the scenario designer. To meet the intended purpose of improving AI routines in military simulations, it would be far better to train it against actual people. One way would be to build this simulator into a two-player machine learning game akin to OpenNERO [21] in which each player trains and evolves a network to serve as their scouts while they control other forces in a simulated battle. For a more passive approach, collecting data on player movement routes over a similar map in a game platform (e.g. in Steel Panthers: Main Battle Tank or observing military units conducting exercises on computer simulations) would provide a wider variety and more realistic sample of enemy movement routes.

Chapter 7: Conclusions

This thesis takes a first look at evolving scout agents using the gas diffusion method to observe enemy units in a simulated environment. In its current state, evolved scout agents consistently outperform randomly placed scouts and they are able to outperform scouts placed by the maximum hex view heuristic in a number of interesting situations. Performance compared to this baseline varies significantly from scenario to scenario, which invites further research to determine what situations evolved scout agents would be the most useful and to determine if there are specific sensor configurations and training regimens which can outperform heuristics on a more consistent basis.

Although there has been previous work in developing computer tools to aid humans in military planning, there has been very little work in developing autonomous agents to exhibit human-like behavior in simulated environments or the real world. This thesis is a first step in exploring that area and lays the groundwork for future and more detailed studies.

References

1. Burgess, Rene' G. *Realistic evaluation of terrain by intelligent natural agents (RETINA)*. Naval Postgraduate School Monterey CA, 2003.

2. Dijkstra, Edsger W. "A note on two problems in connexion with graphs."*Numerische mathematik* 1.1 (1959): 269-271.

3. Duckham, Matt, and Lars Kulik. "“Simplest” Paths: Automated Route Selection for Navigation." *Spatial information theory. Foundations of geographic information science*. Springer Berlin Heidelberg, 2003. 169-185.

4. Headquarters, Department of the Army, Field Manual 1-02, *Operational Terms and Graphics,* Washington, DC, 2004.

5. Headquarters, Department of the Army, Field Manual 3-20.971, *Reconnaissance and Cavalry Troop*, Washington, DC, 2009.

6. Headquarters, Department of the Army, Field Manual 3-20.98, *Reconnaissance and Scout Platoon*, Washington, DC, 2009.

7. Headquarters, Department of the Army, Field Manual 3-90, *Tactics*, Washington, DC, 2001.

8. Headquarters, Department of the Army, Field Manual 3-90.1, *Tank and Mechanized Infantry Company Team,* Washington, DC, 2001.

9. Headquarters, Department of the Army, Army Doctrine Reference Publication 7-0, *Training Units and Developing Leaders*, Washington, DC, 2012.

10. Horn, Graham S., and Jeremy W. Baxter. "An interactive planner for tank squadron assaults." *Proceedings of the 1st planning and scheduling special interest group workshop, Milton Keynes, UK*. 2000.

11. Joy, Bradley, Edward Rykard, and Andrew L. Green. “Integrating Live, Virtual, Constructive Enablers: U.S. Army in Europe’s Ability to Create a Blended Training Environment.” *Armor and Calvary*, April-June 2014. Print.

12. Kenyon, Henry. “Virtual Training Forges Combat Skills.” *Signal* Jul. 2001. *Signal* Web. 1 May 2015.

13. McCullough, Christopher. “Simulated training saves money, prepares Soldiers for combat.” *The Official Homepage of the United States Army,* U.S. Army, 5 May 2014. Web. 1 May 2015.

14. Mora, Antonio Miguel, et al. "CHAC. A MOACO Algorithm for Computation of Bi-Criteria Military Unit Path in the Battlefield." NICSO 2006: 85.

15. Mora, Antonio Miguel, et al. "Enhancing a MOACO for solving the bi-criteria pathfinding problem for a military unit in a realistic battlefield." *Applications of Evolutionary Computing*. Springer Berlin Heidelberg, 2007. 712-721.

16. "Project manager Combined Arms Tactical Trainers (PM CATT)." *U.S. Army Program Executive Office for Simulation, Training, and Instrumentation.* N.p. Web. 1 May 2015.

17. Richbourg, Robert, and Warren K. Olson. "A hybrid expert system that combines technologies to address the problem of military terrain analysis." *Expert Systems with Applications* 11.2 (1996): 207-225.

18. Stanley, Kenneth O., and Risto Miikkulainen. "Efficient evolution of neural network topologies." *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*. Vol. 2. IEEE, 2002.

19. Stanley, Kenneth O., and Risto Miikkulainen. "Evolving neural networks through augmenting topologies." *Evolutionary computation* 10.2 (2002): 99-127.

20. Stanley, Kenneth O., and Risto Miikkulainen. "Evolving a roving eye for go." *Genetic and Evolutionary Computation–GECCO 2004*. Springer Berlin Heidelberg, 2004.

21. Stanley, Kenneth O., Bobby D. Bryant, and Risto Miikkulainen. "Real-time neuroevolution in the NERO video game." *Evolutionary Computation, IEEE Transactions on* 9.6 (2005): 653-668.

22. Stoltenberg, Edward R. "Simulations: Picking the Right Tool for Training." *Cavalry and Armor,* September-October 2012: 41. Print.

23. "Tactical Vignette 97-1: The Battle of Durango Valley." *Armor,* September-October 1997: 37. Print.

24. Vierucci, Ugo. *NEAT Java (JNEAT)*. The University of Texas at Austin Neural Networks Research Group. 24 Jun. 2002. Web. 1 May 2015.

25. winSPMBT: Main Battle Tank. Shrapnel Games, 2015. Web. 1 May 2015.