University of Texas, Austin
Undergraduate Honors Thesis
Supervising Professor: Dr. Risto Miikkulainen

An Analysis of Distributed Decision Making Methodologies in Role Playing Video Games

Matthew Johnston

Abstract

Some environments or tasks may require multiple agents to work together to survive or thrive. Turn-based combat, found in many popular role-playing games, is one such environment. This paper examines a method to expose agents controlled by neural networks to both the solo and group cases in this domain, as well as difficulties presented by the transition from solo to group combat. The results suggest that, dependent on the diversity of agents' skills, agents may choose heterogeneous behaviors regardless of reward scheme.

Introduction

In many situations, individuals may be forced to come together in order to achieve a common goal. Often, individuals in these situations may differ greatly in mind and body, and may naturally specialize within the group. One such situation is the turn-based combat domain, commonly found in role-playing games. Unfortunately, players of these games are often subjected to foes set to a scripted behavior, and may learn to abuse the scripts rather than form a more robust strategy. Additionally, this environment is not limited to one on one combat, and may force groups of players to face off against groups of foes. Since newer, “massively multiplayer” games may require players to repeatedly pit themselves against these groups of foes, they may seem to be a natural fit to expose agents backed by a learning algorithm to a very wide and dynamic variety of human strategies.

This research examines methods to train agents within such an environment, as well as the effects of different reward strategies. Firstly, how do we approach the growth from single participant combat to group combat? If the agents' intelligence is distributed, how do we best reward these agents for acting appropriately within their group? Finally, should rewards be given locally, to individuals who perform well for themselves, or globally, so that every agent in the group is rewarded based solely on the group's success?

Background

rtNEAT

In order to enable learning in the agents used in this research, rtNEAT was used. Developed by Ken Stanley as an extension to the original NEAT (NeuroEvolution of Augmenting Topologies) algorithm, rtNEAT balances the fitness of agents with the diversity of their neural networks, while keeping the evolution and behaviors of agents as continuous as possible. At the end of their evaluation period, a single low fitness rtNEAT agent will be replaced with the offspring of two high fitness agents. Since only one agent is replaced at a time, the behaviors of the population seem continuous. This may be preferable over genetic algorithms that replace

entire populations after each generation, especially in domains that draw heavily from gaming environments.

Cooperative Coevolution & Communication

Much of the work in this paper is informed by observations made by Chern Han Yong, as well as the background for his work. Yong[2] found that agents with distributed intelligence may be more effective than those directed by a centralized controller. Furthermore, as hinted by Franklin[3], Yong[2] found that role-based behaviors may offer more advantageous behaviors than those developed using communication. This paper expands on the development of role-based behaviors in a domain where agents may have heterogeneous bodies and abilities.

Finally, Balch's[4] conclusion on the effects of local and global rewards are examined within the turn-based combat domain. This paper's domain differs from Balch's[4] by offering heterogeneity between the bodies and abilities of agents within a team. This allows the learning agents to differ not simply in their networks, but in physical capabilities as well. This key difference may substantially alter possible solutions within the domain.

Turn-based Combat Domain

Agents

In order to define how combat takes place within the domain used in this research, it is necessary to first describe how the agents exist within their environment. Each agent within the environment belongs to a class. Classes describe an agent's body at its peak, denoting what actions an agent is capable of and how efficient it is at those actions. Agents within the environment may have heterogeneous classes, allowing for variations not only in the agents' learning but also in physical capabilities. Since classes could be made with unique abilities in mind, an agent's possible role may be suggested by its class.

Class Statistics

Health	Represents the maximum amount of punishment an agent of this class can withstand. Once the agent's health drops to zero, the agent falls.
Mana	Represents the maximum amount of energy reserved for special actions.
Strength	Denotes the class's ability to deal physical damage.
Vitality	Denotes the class's ability to mitigate physical damage.
Dexterity	The class's accuracy, as well as speed. More accurate agents may strike critical blows.
Intelligence	Denotes the class's ability to deal magical damage.

Willpower	Denotes the class's ability to mitigate magical damage, and resist status effects.

Agents within the environment are placed into two groups. The first group, the “party”, is a group of learning agents, working with each other for two purposes. The first purpose is survival, followed by the harm of the second group. The party is made of a set of positions, with each position being reserved for a certain class of agent. Combat ends in failure when any of the agents in the active party fall in battle. The second group is a group of dummy agents, fodder for the learning agents to train against. This group of fodder will never truly die, but their damage is still recorded to measure the party's offensive success.

Combat

Once agents are defined within the domain, we can describe how they interact within it to achieve their goals. In order to affect the state and flow of combat, agents take turns acting in battle. Each agent is given an individual readiness “container” statistic, much like their individual health and mana statistics. At the beginning of combat this statistic is set to 0, and will fill to 100 at a speed governed by the agent's class dexterity. When an agent's readiness is full, it becomes available to take an action. If at any time during combat no agents are available to act, every agent will increment their readiness. Combat will end after the agents have collectively consumed 1000 turns. The party is considered to have a successful battle if all of its members survive until no turns are left. Since speed is based on dexterity, agents of more dextrous classes will have more opportunities to act than others.

If the turn structure is the beating heart of the turn-based combat domain, then actions comprise the mental and strategic aspects. Actions are the sole mechanisms the agents have at their disposals to alter and control the state of combat. Actions vary widely in effect, as well as in exclusivity between classes. Once it is available to act, an agent takes its turn by choosing an action and a specific target to perform the action on. After an agent performs an action, its readiness is reset to 0. While some turn-based combat domains allow friendly fire or other similar strategies, the domain used here does not. The “hostility” of an action determines the group targeted by the action, then the agent chooses a specific target within that group. Finally, an action is considered to be successful only if it alters the state or flow of combat. For example, an action that applies a negative status effect such as “sleep” will only be successful if the target is actually put to sleep.

Friendly Actions

Cure	Restores a portion of the target's health at the cost of mana. Increases in effectiveness with willpower.
Cover	Places the user in a defensive position in front of the target, taking damage in their place.
Refresh	Applies the “refresh” status effect at the cost of mana, restoring a small amount of mana to the target every turn.
Haste	Applies the “haste” status effect at the cost of mana, doubling the speed of which the target's readiness is filled.
Blink	Places a protective barrier around the target at the cost of mana. This barrier will absorb a single hostile action.

Hostile Actions

Attack	Causes physical harm to the target, reducing their health. Increases in effectiveness with user's strength, decreased by target's vitality.
Fire	Causes magical harm to the target at the cost of mana. Increases in effectiveness with user's intelligence, decreased by target's willpower.
Poison	Applies the “poison” status effect at the cost of mana, causing slight harm to the target every turn.
Sleep	Applies the “sleep” status effect at the cost of mana, effectively reducing the target's readiness to -100. The target is considered awake when readiness becomes positive. Any action performed on a sleeping target will wake them.
Reap	Causes great physical harm to the target at the cost of the user's own health.

Soloing

Many issues in the turn-based combat domain are simplified in solo combat. The first of these is the representation of the environment that the agent sees. Since the agent needs to know if it should heal itself, it needs to know its own health. In addition, the agent also needs to know how much mana it has left, so that it does not fail to use an ability that requires it. Statistics such as strength and intelligence will never change, so are unnecessary for the agent to see. Since the enemy is essentially immortal, his health and mana are unimportant to the learning agent and are omitted from the inputs.

Targeting is similarly simple, even more so when friendly fire and stray healing are disallowed. Given that constraint, the agent's target is decided by the hostility of his action. Cure, being a friendly or helpful action, will always target the agent using it, while destructive actions such as Attack and Sleep are typically hostile and will target an opponent. Keeping with the pattern, local rewards are easily decided and given appropriately. Any action performed in combat can very easily be traced back to the only possible user.

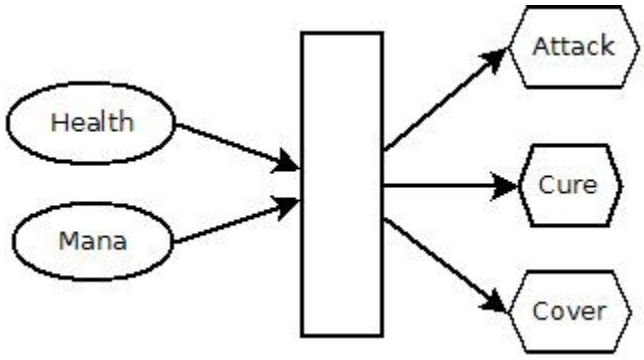

Figure 1: 1v1 Simplified neural network structure, from a Paladin's viewpoint. Here the Paladin only needs to choose an action to perform.

Groups

With the addition of more agents into the fray, the turn-based combat domain becomes far more complicated. Since a victory for the party is based on the survival of all of its members, it becomes a necessity for the agents to ensure the longevity of their comrades. Teamwork becomes far more meaningful and difficult to reward locally. Some actions, such as Blink or Cover, cause no direct numerable change to the flow of battle, and therefore may be difficult to reward directly without doing so arbitrarily. Agents with classes that naturally tend to use these actions may form a foundation for agents that are inclined to more obviously “fit” strategies. Finally, targeting becomes an issue. In this case, agents require a mechanism that allows them to not only choose a friendly or hostile action, but to choose which member of the friendly or hostile group to use the action on.

In order to assist their allies in combat, members of the learning party may need a method of evaluating their allies' health and mana, as well as any status effects. Since the actions used for this research limit the ability to remove status effects, they will not be necessary in this simplified domain. This may cause the particular domain to become a role-based cooperative one, as defined by Yong[2], but more advanced versions may allow for situations that may require communication to solve.

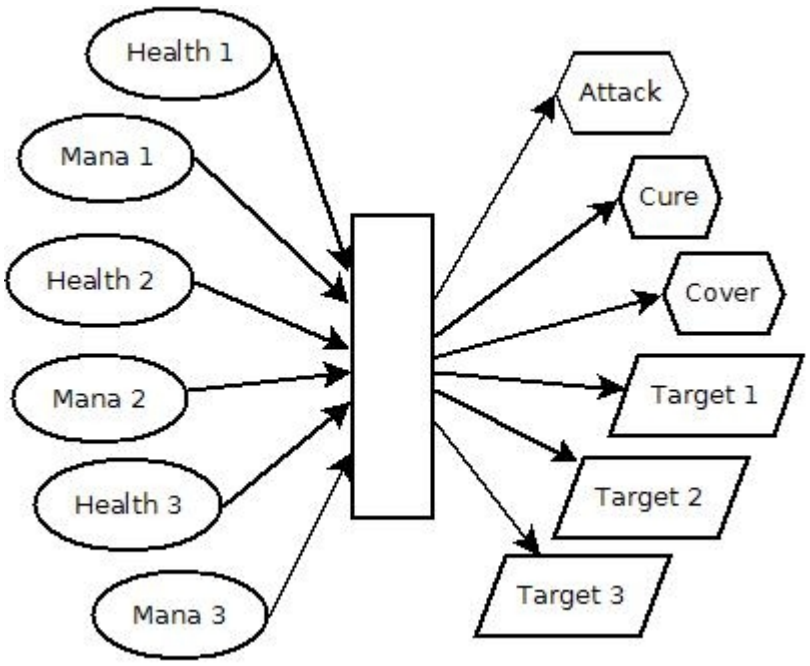

Figure 2: 3v1 Simplified neural network structure, from a Paladin's point of view. Here the Paladin must choose an action-target pair in order to act.

Experiments

In order to address the many questions and issues put forth in the introduction of this paper, a series of experiments were run. The first was a simple test of one versus one combat, to ensure that the platform developed for this research was appropriate in the solo combat case, before expansion to multiple combatants. The second experiment tested the platform in the group case. The final experiment, intended to test local and global reward structures, was spread over 16 sets of data with different party configurations, containing 1500 battles each. Each position in the party was given a population of 50 rtNEAT agents to train.

Experiment 1: Solo Combat

The initial experiment was intended as a proof of concept for the solo combat case. As mentioned above, the agent used set the standard for average statistics, and his speed was set to match the opponent's. To act, the agent was given the ability to use Attack, Cure, Fire, Poison, and Reap, as well as a dummy action, Idle. As Idle essentially wastes the agent's turn, the agent was expected to quickly learn that it should never be used. The hope was that after Idle was abandoned, the agent would form a somewhat decent damage strategy. Since the agent was given no capability of restoring his mana, it was inevitable that it would fall in combat.

The agent's tremendous success at maximizing his damage output was occasionally surprising, but mostly expected. It learned fairly quickly to use Cure when closer to death. It learned far less quickly to use Poison very early in a fight. Fire was generally abandoned unless the damage output of the action was altered to be ridiculously overpowered. Presumably, the cost of using mana for Fire was not worth losing the turns gained by using Cure. While generally ignored, the agent would sometimes learn to make use of Reap on the verge of death. This sort of desperate strategy is reminiscent of kamikaze pilots and suicide bombers, and while to some degree it was intended, it still gives the author pause.

Experiment 2: Expansion 2v1, 3v1

After the success of the initial experiment, it was necessary to show this paper's approach can produce appropriate and varied solutions for group combat in its domain. To do this, the approach was tested on a variety of agent classes and party configurations, in keeping with the expectation in many role-playing games. The classes used were as follows:

- Paladin – A stereotypical defensive class.
  - High maximum health, low maximum mana
  - High vitality and willpower, average strength, lacking in intelligence and dexterity
  - Able to use Attack, Cover, and Cure
- Sorcerer – A supporting class.
  - Average maximum health and mana
  - Extremely dextrous, average intelligence and willpower, lacking strength and vitality
  - Able to use Attack, Refresh, Haste, and Blink
- Magician – A magically offensive class, fragile.
  - High maximum mana, low maximum health
  - High intelligence and willpower, average dexterity, lacking in strength and vitality
  - Able to use Attack, Fire, Poison, and Sleep
- Onion Knight – A stereotypically average class. Jack of all trades.
  - Average maximum health and mana
  - Average strength, vitality, dexterity, intelligence and willpower
  - Able to use every action available within the domain

Eight data sets used two versus one configurations, and the other eight used three versus one. Of the two versus one configurations, four configurations contained a Paladin and Sorcerer, and the remaining contained two onion knights. The three versus one configurations were similar, the heterogeneous party setup gained a Magician, while the homogeneous gained

an additional onion knight. Every configuration pitted the party against a single fodder agent, capable only of using Attack on a random target.

The results varied greatly depending on reward scheme and classes used. While every configuration produced a survival solution, the agents within a few configurations were unable to converge to it after discovery. This shows the possibility for rtNEAT to discover appropriate survival solutions within the domain, despite some issues. In general, the solutions found used "spamming" strategies. After an initial setup phase, agents would often repeat the same action until combat had ended.

Example solutions found included:

- 2v1 Spam Cure
  - The Paladin covers the Sorcerer, then cures himself repeatedly
  - The Sorcerer refreshes the Paladin repeatedly
- 2v1 Spam Blink
  - The Paladin covers the Sorcerer
  - The Sorcerer refreshes himself, then blinks the Paladin repeatedly
- 3v1 Spam Sleep
  - The Paladin continuously attacks
  - The Sorcerer continuously refreshes the Magician
  - The Magician continuously uses sleep

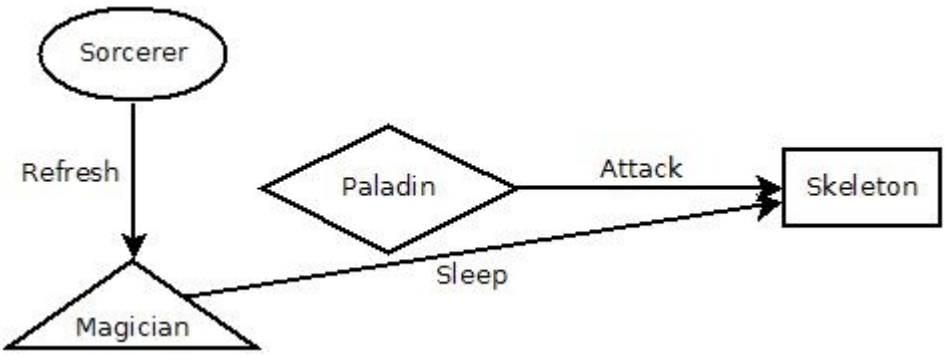

Figure 3: Representation of Spam Sleep strategy

While spam solutions are certainly legitimate, they may suggest the development of role-based behavior. This is fine for classes used within this paper's simplified domain, but could be a problem for more sophisticated domains requiring communication.

Experiment 3: Local vs. Global Rewards

Confident that the approach used in this paper was successful in finding solutions in the group case, an experiment was made to test the effects of different reward schemes. The

experiment was intended to test the speed of finding solutions, as well as the solution types produced. Each party configuration described in experiment 1 was given 1500 battles to find a solution, first with local rewards, then with global. Local rewards were given based on numeration from damage and healing output, as well as for small, arbitrary amounts of "karma" for successful actions lacking any direct numeration. Global rewards were given to the whole party based on the sum of turns survived and the number of times the fodder agent's health was reduced to zero.

While the results whether local or global rewards converge to a solution more quickly were inconclusive, the stability introduced by each reward scheme became apparent. Of the eight local reward configurations, five converged to a single stable solution within 1500 battles. The remaining three, all homogeneous class configurations, never saw the agents converge to a single solution, but did converge to a set of solutions. Of the eight global reward configurations, half converged to a a single stable solution. The remaining half not only failed to converge to a single solution, but consistently reverted to strategies that proved fatal to the party.

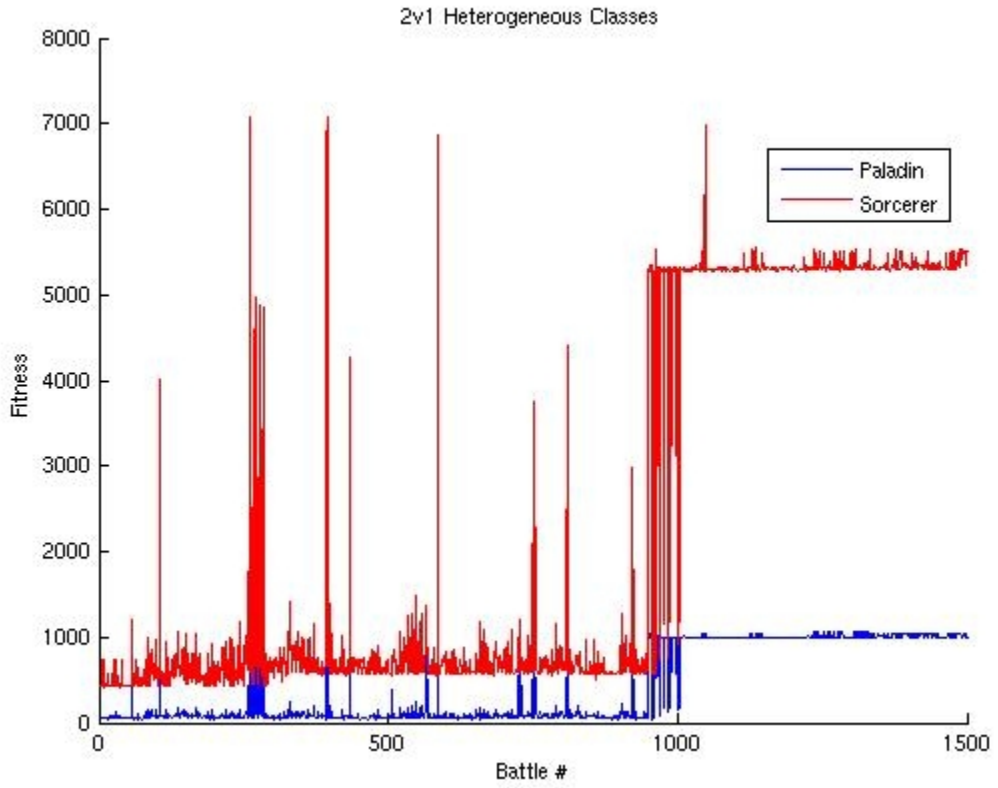

Figure 4: In the local reward case seen here, the heterogeneous 2v1 configuration finds and converges to a survival solution. The Paladin and Sorcerer can be seen to converge to separate roles.

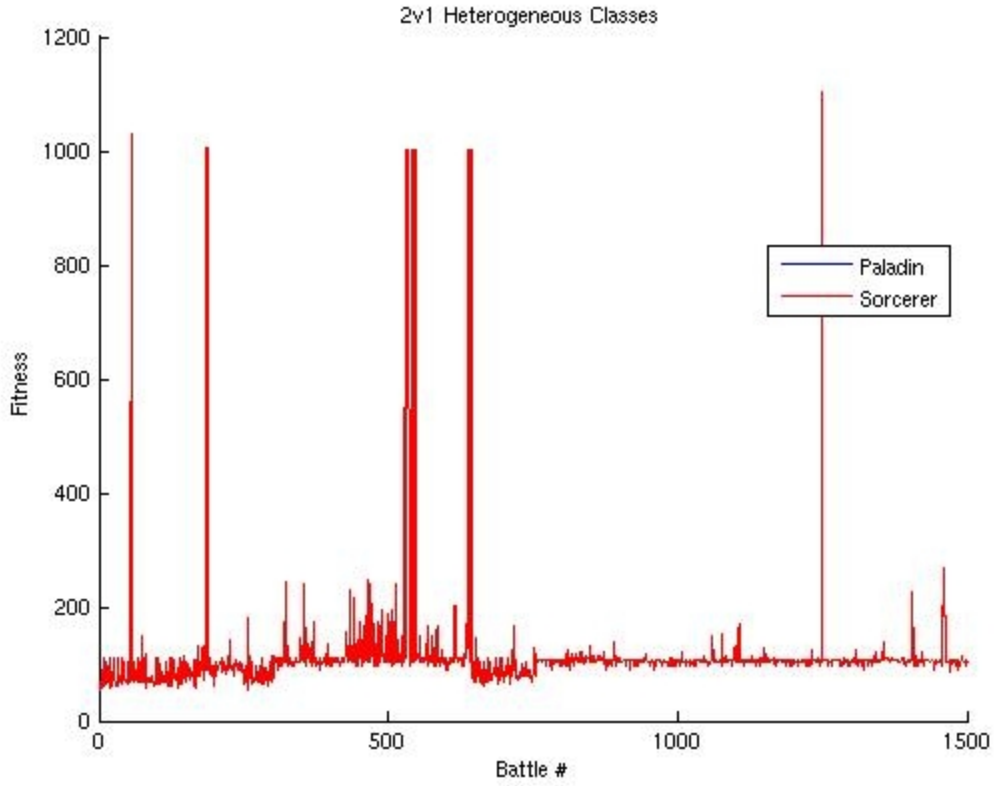

Figure 5: In the global reward case, the heterogeneous 2v1 configuration finds a survival solution at least six times, but is unable to converge to it. Worse, it reverts to behaviors that kill the party.

Regarding behavior, the results indicated that local rewards did not always produce identical behaviors across party members. In the heterogeneous party case this disagreement with Balch[4] is not surprising, since the agents may not have the physical capacity to perform the same as their peers. However, in the homogeneous case the answer is less clear. An answer may lie in the popularity of the "Spam Sleep" strategy. Agents are rewarded for sleep only if the action successfully puts the target to sleep. If the target is already sleeping, then the action is a waste and is not rewarded. If two agents are sleep spamming the same target at once, only the first agent will actually receive the reward. In agreement with Balch[4], global rewards also produced heterogeneous behaviors. However, this may again be an artifact of the of the difference between agent's classes, as well as reward subtleties such as the sleep case.

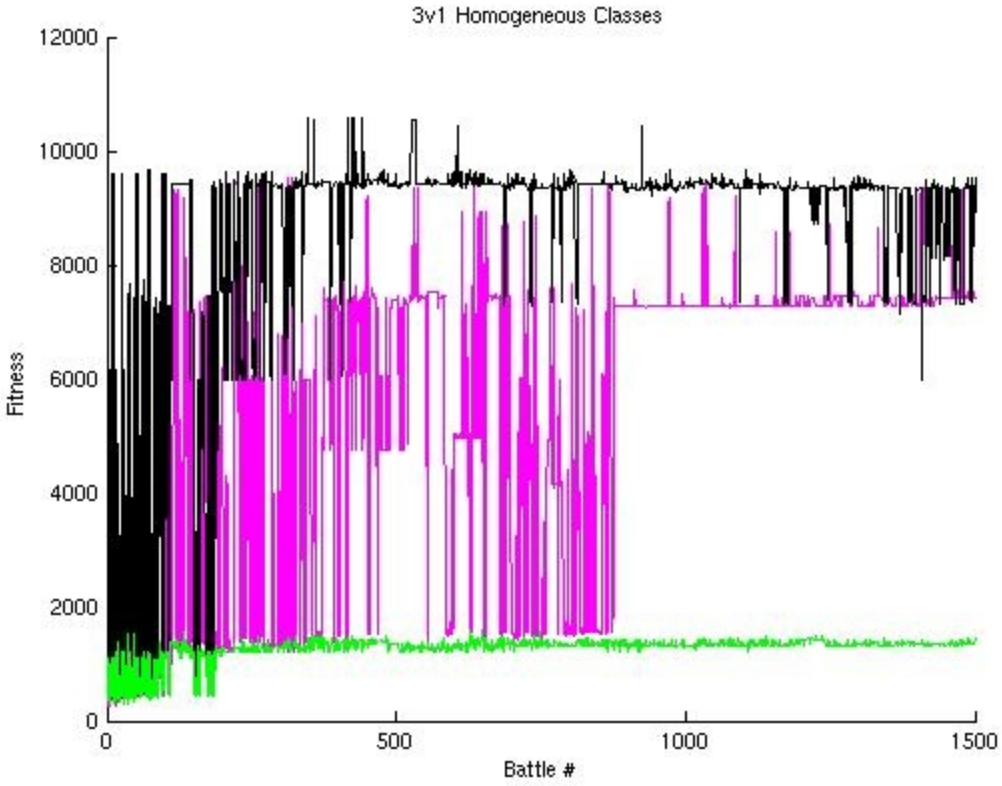

Figure 6: In the local reward case seen here, 3 homogeneous agents compete for dominant roles.

Conclusion

In group domains where agents have identical classes, it has previously been shown that local rewards lead to identical behaviors and global rewards lead to diverse behaviors. However, the results in this paper suggest that agents with heterogeneous classes may develop diverse behaviors despite local rewards. In addition, some subtleties in the domain's reward structure may prevent agents with homogeneous classes from developing identical behaviors, regardless of local rewards.

Finally, agents within the turn-based combat domain used in this paper highly favored role-based behaviors over communicative ones. While an agent's class may suggest the role it will eventually form, even homogeneous classes will form roles in this domain.

References

[1] Stanley, K., Bryant, B.., Karpov, I. and Miikkulainen, R. (2006). Real-Time Evolution of Neural Networks in the NERO Video Game.

[2] Yong, C. and Miikkulainen, R. (2007). Coevolution of Role-Based Cooperation in Multi-Agent Systems.

[3] Franklin, S. (2001). Coordination without communication.

[4] Balch, T. (1997). Learning roles: Behavioral diversity in robot teams.