Effects of Shared Perception on the Evolution of Squad Behaviors

Darren Doherty and Colm O'Riordan

***Abstract*—As the nonplayable characters (NPCs) of squad-based shooter computer games share a common goal, they should work together in teams and display cooperative behaviors that are tactically sound. Our research examines genetic programming (GP) as a technique to automatically develop effective squad behaviors for shooter games. GP has been used to evolve teams capable of defeating a single powerful enemy agent in a number of environments without the use of any explicit team communication. This paper is an extension of our paper presented at the 2008 Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE'08). Its aim is to explore the effects of shared perception on the evolution of effective squad behaviors. Thus, NPCs are given the ability to explicitly communicate their perceived information during evolution. The results show that the explicit communication of perceived information between team members enables an improvement in average team effectiveness.**

***Index Terms*—Evolutionary computation, genetic programming, shooter games, squad behaviors.**

I. Introduction

In recent years, there has been an emergence of squad-based shooter games. The AI of the nonplayable characters (NPCs) of these games should be team-orientated and tactical as the NPCs should work together to devise the most effective method to achieve their common goal. As tactics are highly dependent on the situation (i.e., team supplies, enemy movement, etc.) [2], it is very difficult for developers to code the tactical team behaviors and decide when and where it would be effective to use certain tactics. As such, developers find it difficult to create squads of NPCs that are able to correctly assess a situation, choose effective courses of action for each NPC, and work together to achieve their common goal.

Rather than attempting to develop complex behavioral systems that may allow NPCs to display intelligent squad behavior, game developers have opted to continue using rudimentary techniques to implement the AI of individual NPCs and use simple techniques to make it appear as if the NPCs are cooperating in an intelligent manner. For example, some developers prevent two NPCs from simultaneously shooting at the player, causing them to appear to be taking turns attacking. This is combined with audio cues from the NPCs such as shouting "cover me" when an NPC reloads its weapon to create the illusion of cooperative behavior. Using "cheating" mechanisms to simulate cooperative behavior has been successful in giving the appearance of intelligent, cooperative agents. However, using more sophisticated AI to create teams that are actually intelligent and cooperative rather than teams that appear so is an important research question, not only in computer games but also in other domains. Using sophisticated AI could also create a more challenging gaming experience as it may allow teams more advanced reasoning abilities than current approaches.

We propose that genetic programming (GP) can be used to evolve effective squad behaviors for shooter games. GP has not yet been used in commercial computer games and its potential usage in computer games has not been researched to any extent in academia. Despite this, GP has been very successful at evolving team behaviors in a variety of simulated environments [3]–[9] (see Section II). GP is also considerably flexible as it allows programmers to develop more diverse solutions than those provided by genetic algorithms (GAs) and an evolved genetic program is directly interpretable by humans. The goal of this paper is to show that GP can be used to evolve effective squad behaviors for shooter games. However, GP could also be used to create "interesting" or "fun" NPC behaviors by altering the manner in which fitness scores are evaluated. The level of "fun' associated with an NPC is a subjective matter and hence the evaluation could possibly involve human feedback following the evolution of "fit" solutions. Alternative approaches to evolving "fun" behaviors would be to constrain the evolution to only generate solutions that are similar to those identified as interesting by users. This represents a separate question to the one being addressed in this paper. We are primarily interested in generating fit solutions to illustrate the effectiveness of GP in this regard and to gain a better insight into the complexity of the problem. The evolution of "fun" behaviors represents, for us, a separate question which can be more fully tackled once the first question regarding the evolution of "fit" behaviors has been answered.

In our previous research, GP has been successfully used to evolve effective teams in shooter environments of varying difficulty [10]. Teams are evolved against a single powerful enemy agent that can be likened to the human player of a single-player shooter game. The difficulty of the environment is varied by altering the field of view (FOV) and viewing distance of NPCs. In modern shooter games, both the NPCs and the human player(s) have limited visual ranges within which information can be perceived. In [10], the evolved teams could only implicitly communicate by observing the positions of other agents. In this paper, NPCs are given the ability to share perceived information in order to explore the effects of perceptual communication

Manuscript received November 14, 2008; revised January 08, 2009; accepted March 04, 2009. First published March 24, 2009; current version published May 01, 2009.

The authors are with the Department of Information Technology, National University of Ireland Galway, Galway, Ireland (e-mail: darren.doherty@nuigalway.ie; colm.oriordan@nuigalway.ie).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCIAIG.2009.2018701

on the evolution of effective squad behaviors. In order to compare teams evolved with perceptual communication to teams evolved without perceptual communication, the gaming environments and genetic program used in this research are identical to those used in [10]. We hypothesize that explicit communication should allow the NPCs to perceive the environment as a team rather than individually, which should result in more effective emergent squad behaviors.

II. Related Work

With the emergence of squad-based shooter games, developers have struggled to create systems that allow teams of NPCs to display effective squad behaviors. As such, developers have opted to use simple techniques to create the illusion of cooperation among the NPCs. Command hierarchies [11] and cognitive architectures [12] have both been proposed as methods to implement squad AI for shooter games. Decentralized approaches [13] where the team behavior emerges from interactions of team members and centralized approaches [14] where a team leader makes the decisions have also been suggested. However, there is no consensus on which approach is best and all require a considerable amount of time and effort to design and implement. Although our proposed GP approach also takes time to initially setup, once implemented, it can potentially be used to create squad behaviors to perform numerous tactics in a range of different environments.

Evolutionary computation (EC) techniques have not been used extensively in the exploration and research of AI for computer games. As the environments and range of NPC behaviors in a computer game are generally very complex, developers are hesitant to introduce EC techniques into their games as there is no guarantee desirable behaviors will be found. EC techniques are more suited to offline game AI development (i.e., during game development) as opposed to online (i.e., evolving while the game is being played to allow the NPCs to adapt in real time) as they are very resource intensive and can take a long time to produce solutions. The research community has begun to realize the potential of EC techniques as developmental tools for game AI. Champandard [15] used a GA to successfully evolve NPCs in a first-person shooter game to dodge enemy fire. Ponsen [16] used a GA to successfully design tactics for a real-time strategy (RTS) game and Cole *et al.* [17] used a GA to tune an NPC's weapon selection parameters for a shooter game. In addition, GP has been used to evolve behaviors for simple computer games, such as PacMan [18], Tic-Tac-Toe [19], Tetris [20], and Snake [21].

A few attempts have been made at evolving teams for shooter games [22], [23]. The first system [22] uses neuroevolution to evolve teams but requires a human player to specify which attributes are to be evolved, and the second mechanism [23] uses an adapted GA representation that requires a number of game-specific enhancements to the GA paradigm. Both techniques have been used to successfully evolve squad behaviors. However, neither technique is ideal for developers to use to create squad behaviors for NPCs. In both cases, the team's behavior is evolved in an adaptive manner, while the game is being played, so developers cannot tell or test, in advance, what behaviors the NPCs will exhibit or how tactically proficient they will be. Online reinforcement learning (RL) techniques have also been suggested to allow squads in shooter games to adapt [24], [25]. As EC techniques are more suited to offline AI development, our research is concerned with offline evolution rather than online adaptation of team behaviors.

GP has been successfully used to simulate team evolution in a number of different simulated domains. GP was first applied to team evolution by Haynes *et al.* [26]. With respect to team evolution, GP has been mainly used to solve multiagent control problems using teams of cooperating agents. Pursuit strategies for predator–prey domains have been successfully evolved using GP techniques [3], [4]. Haynes *et al.* [3] used a strongly typed genetic program [27] to evolve a team of predators to hunt a single prey and Luke and Spector [4] used GP to successfully evolve predator strategies that enable a group of lions to successfully hunt faster moving gazelle across a toroidal, continuous 2-D landscape. In Luke's work, heterogeneous teams are shown to perform better than homogeneous teams. In addition, Reynolds used GP to evolve prey that exhibit a herding behavior when confronted by predators [5]. GP has also been used to enable a team of ants to work together to solve a food collection problem [6]. The ants must not only cooperate in order to reach the food but must also work together to carry it as it is too heavy for one ant to carry alone. Richards *et al.* [7] used a genetic program to evolve groups of unmanned air vehicles to effectively search an uncertain and/or hostile environment. Their environments were relatively complex, consisting of hostile enemies, irregular shaped search areas, and no-fly zones. In addition, GP has been used to successfully evolve sporting strategies for teams of volleyball players [8] and teams of soccer players [9].

It has been argued that communication is a necessary prerequisite to teamwork [12] and plays a key role in facilitating multiagent coordination in cooperative and uncertain domains [28]. Moreover, in a study conducted by Barlow *et al.* [29] on teamwork in shooter games, it was found that communication is one of the three main factors that contribute to a team's success, together with role assignment and coordination.

III. Gaming Environment

The environment is a 2-D space, enclosed by four walls and is built using the *Raven* game engine[1] [30, ch. 7]. Items are placed on the map at locations equidistant from both the team and enemy starting points. These items consist of health packs and a range of weapons that respawn after a set time if collected (see Fig. 1).

Both types of agent (i.e., team agent and enemy agent) use the same underlying goal-driven architecture [30, ch. 9] to define their behavior. The goal-driven architecture uses a hierarchy of goals in which composite goals are broken down into subgoals. Goals are satisfied consecutively so the current goal (and any subgoals of it) is satisfied before the next goal is evaluated. If an NPC's situation changes, a new, more desirable goal can be placed at the front of the goal queue. Once this goal is satisfied, the NPC can continue pursuing its original goal. Although the

[1]The *Raven* game engine source code can be downloaded from http://www.wordware.com/files/ai/Buckland_AISource.zip

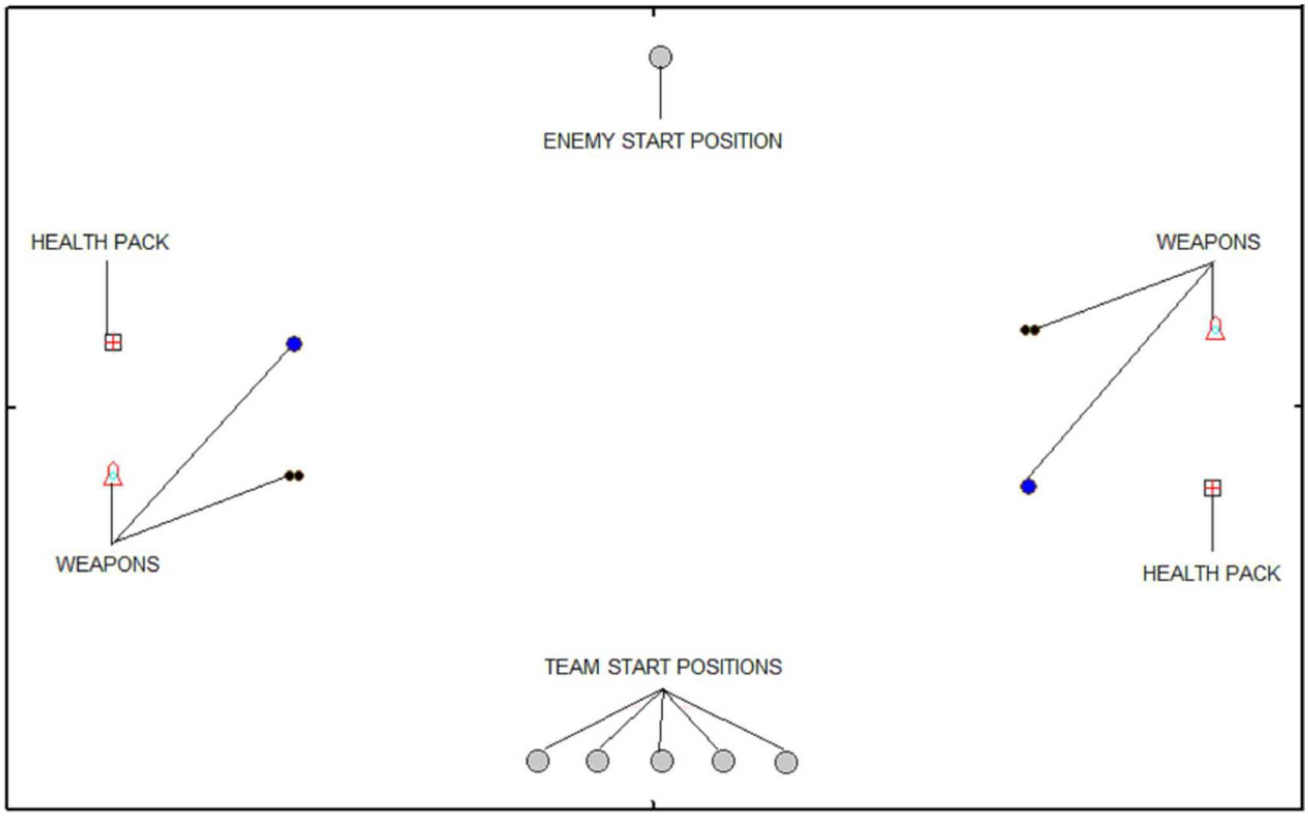

Fig. 1. Environment map.

underlying goal architecture is the same, team agents use a decision-making tree evolved using GP to decide which goal to pursue, whereas the enemy uses desirability algorithms associated with each goal. These desirability algorithms are hand-coded to give the enemy intelligent reasoning abilities. Random biases are used when creating these desirability algorithms, in order to vary the enemy's behavior from game to game. These desirability algorithms coupled with the random biases define a range of rational and intelligent decision-making abilities that mimic how a human might behave when playing the game. A detailed description of the desirability algorithms is given in [10].

The team consists of five agents each of which begins the game with the weakest weapon in the environment. The enemy agent has five times the health of a team agent and begins the game with the strongest weapon with unlimited ammunition. Both types of agent have a memory allowing them to remember information they perceive. Any dynamic information, such as team or enemy positions, is forgotten after a specified time. All agents have limited visual and auditory ranges within which they can perceive information in their environment. An agent's visual capability is defined by their FOV and viewing distance. For these experiments, the enemy can see twice as far as team agents but the FOVs of both agent types are equal. Auditory ranges are the same for all agents. If more than one team agent has been recently sensed by the enemy, the enemy will select its target based on distance. Weapons have different firing frequencies and ideal ranges within which they are more effective and bullets have different properties, such as velocity, spread, damage, etc.

IV. The Genetic Program

The genetic program structure, GP nodes, and GP operators used in these experiments are identical to those used in [10] so that the results can be compared. A complete list of GP nodes, together with parameters used in the evolution can be found on IEEE Xplore. In our genetic program, the entire team of five NPCs is viewed as one chromosome, so team fitness, crossover, and mutation operators are applied to the team as a whole. Each agent is derived from a different part of the chromosome, so evolved teams are heterogeneous (see Fig. 2).

A strongly typed genetic program is used so as to constrain the type of nodes that can be children of other nodes. In strongly typed GP [27], the initialization process and genetic operators must only allow syntactically correct trees to be produced. There are five types of node and a total of 50 nodes used in the evolution.

- Action nodes represent goals the agent can pursue and the *IF* statement.
- Condition nodes represent conditions under which goals are to be pursued.
- Position nodes represent positions on the map (relative to other agents) to which the agents can move
- Environment nodes represent gaming parameters that are checked during the agent's decision making process (e.g., ammunition supplies).
- Number nodes represent arithmetic operators and constants.

Note that the *IF* node has an arity of three. The first child node is a condition node which evaluates to true or false. If

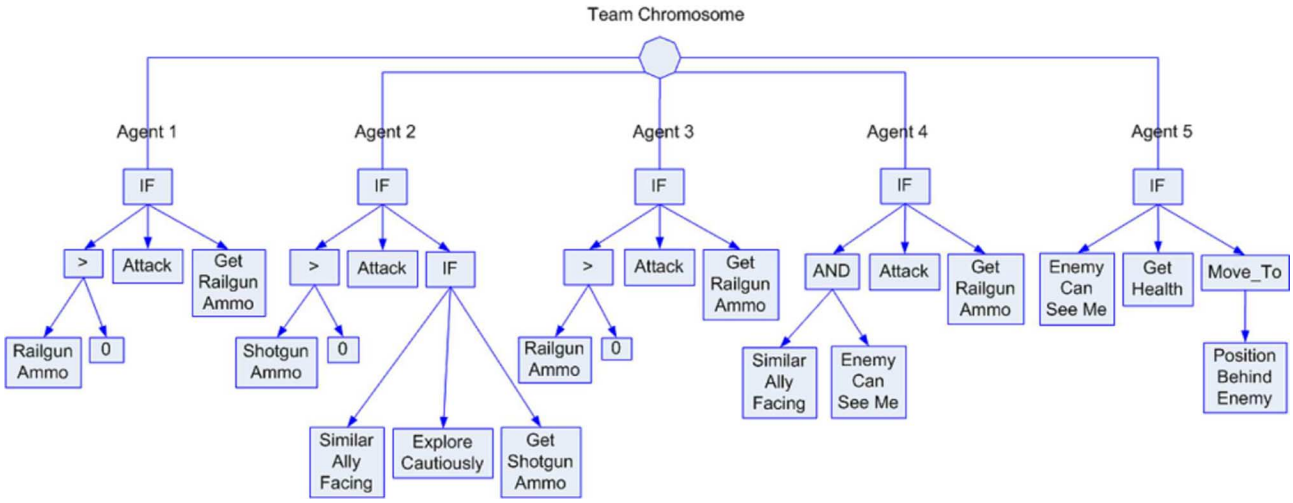

Fig. 2. Sample evolved GP chromosome.

true, the second branch is followed, otherwise the third branch is followed. These branches contain nodes representing goals the agent should pursue or further *IF* nodes (see Fig. 2).

A. Fitness Calculation

The fitness function takes into account the games' duration and the remaining health of the enemy and team agents

$$\text{RawFitness} = \frac{\text{AvgGameTime}}{\text{Scaling} \times \text{MaxGameTime}} + \frac{\text{EW} \times (\text{Games} \times \text{TSize} \times \text{MaxHealth} - \text{EH}) + \text{AH}}{\text{Games} \times \text{TSize} \times \text{MaxHealth}}$$

where AvgGameTime is the average duration of the games, Scaling reduces the impact that game time has on fitness (set to four), MaxGameTime is the maximum game length (i.e., 5000 game updates), EH and AH are the amount of health remaining for the enemy and for all five team agents, respectively, EW is a weight that gives more importance to EH (five), Games is the number of games per evaluation (i.e., 20), TSize is the team size (i.e., five), and MaxHealth is the maximum health of a team agent (i.e., 50).

The team's fitness is then standardized such that values closer to zero are better and the length of the chromosome is taken into account to prevent bloat

$$\text{Fitness} = (\text{MaxRF} - \text{RawFitness}) + \frac{\text{Length}}{\text{LengthFactor}}$$

where MaxRF is the maximum value RawFitness can hold, Length is the length of the chromosome, and LengthFactor is a constant used to limit the influence Length has on fitness (set to 5000).

B. Selection

There are two forms of selection used. The first is a form of elitism where m copies of the best n chromosomes from each generation are copied directly into the next generation. Three copies of the best and two copies of the next best individual are retained in this manner. The second method is roulette wheel selection. Any chromosomes selected in this manner are subjected to crossover and mutation (given probabilities of 0.8 and 0.1, respectively). To increase genetic diversity, there is also a 2% chance for new chromosomes to be created and added to the population each generation.

C. Crossover

The crossover operator is specifically designed for team evolution [31]. At the start of each crossover operation, a random TSize bit mask is selected that decides which of the agents in the parent chromosomes are to be altered during crossover. A "1" in the mask indicates that the agent at that position is copied directly into the child and a "0" indicates the agent is to take part in crossover with the corresponding agent of the other parent (see Fig. 3). A random crossover point is then chosen within each agent to be crossed over. The node at the crossover point in each corresponding agent must be of the same node type in order for the crossover to be valid.

D. Mutation

Two forms of mutation are used in these experiments, one to allow good subtrees to spread within a team and the other to help maintain diversity in the population. The former mutation method, known as intrateam mutation, randomly chooses two agent trees from the same team chromosome and swaps two randomly selected subtrees between the agents (see Fig. 4). Similar to the crossover operation, the root nodes of the subtrees must be of the same node type for the mutation to be valid. The second form of mutation, known as swap mutation, randomly selects a subtree from the chromosome and replaces it with a newly created random tree (see Fig. 5).

V. Phenotypic Analysis

The phenotypic analysis technique used in this work is based on our earlier approach to phenotype analysis presented in [32]. In order to analyze the phenotypes of evolved team chromosomes, we first define a method to capture the behavioral information of each team agent from the gaming environment. Our phenotypic analysis technique uses this behavioral information to determine the behavioral characteristics each agent on the team is displaying. These behavioral characteristics are

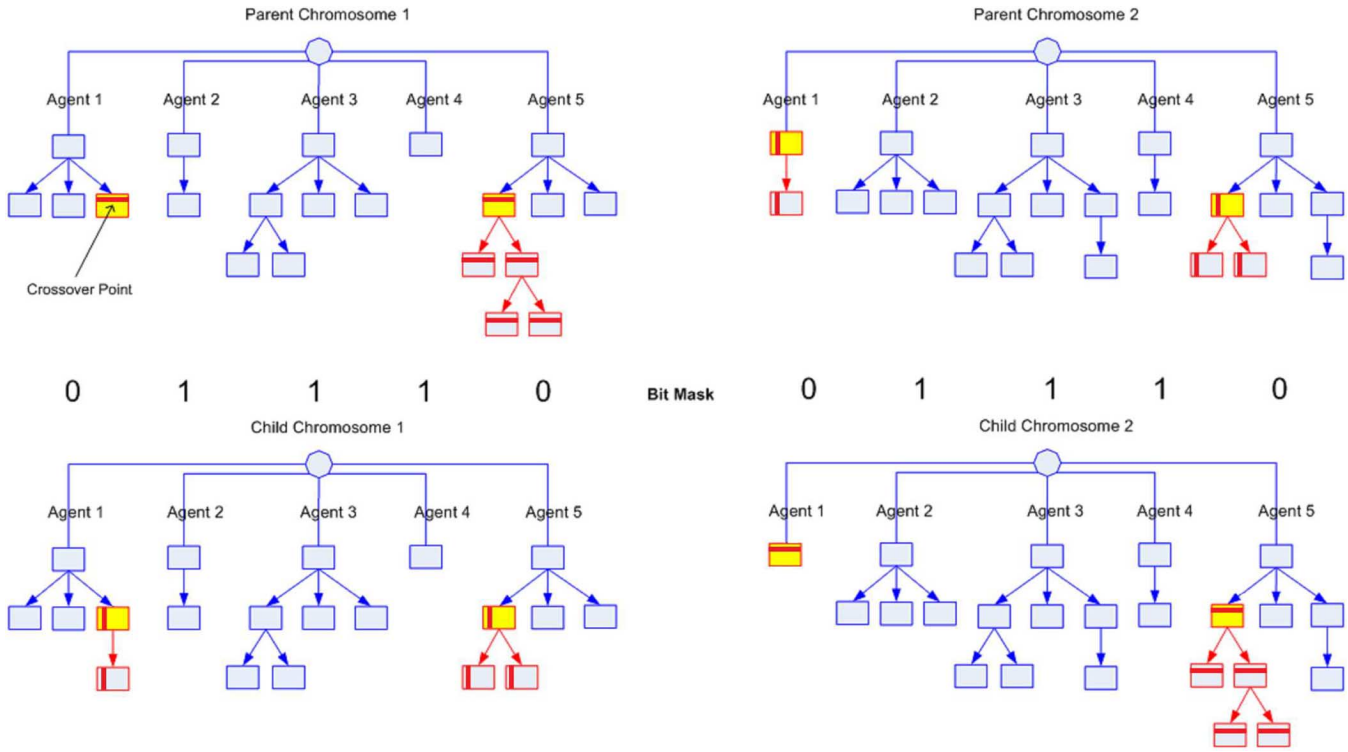

Fig. 3. Sample crossover operation between two team chromosomes.

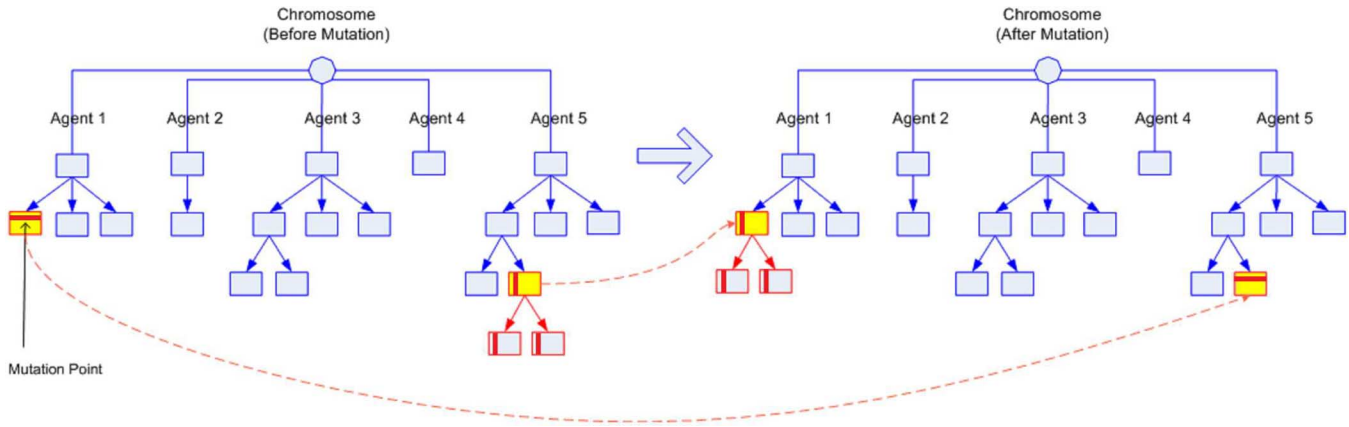

Fig. 4. Sample intrateam mutation operation between two team agents.

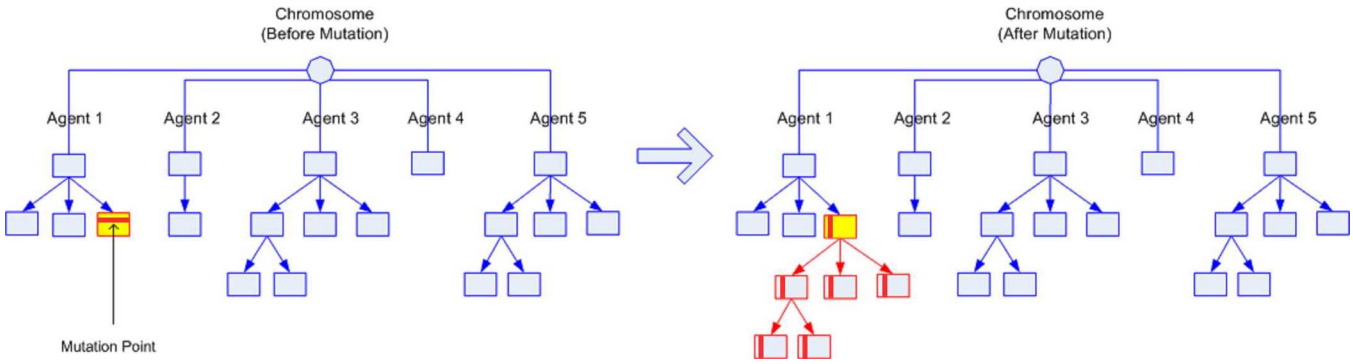

Fig. 5. Sample swap mutation operation.

TABLE I
BEHAVIORAL INFORMATION CAPTURED FOR TEAM AGENT

Game Number	1	2	3	4	...
Railgun Damage Inflicted	110	80	100	50	...
Shotgun Damage Inflicted	0	0	0	0	...
Rocket Damage Inflicted	0	0	0	0	...
Blaster Damage Inflicted	11	2	4	0	...
Average Health	47	50	48	24	...
Distance To Ally1	1485	1587	1386	1973	...
Distance To Ally2	9540	8563	8848	9867	...
Distance To Ally3	14000	14102	14185	15087	...
Distance To Ally4	14870	14982	14857	14887	...
Distance To Enemy	5250	4580	4478	5870	...
Enemy Target	5	8	4	17	...
Lifetime	30	19	20	15	...
Game Length	30	19	20	28	...
Survived Game	1	1	1	0	...

encapsulated in a number of different agent types that have been identified arbitrarily from observation. In this work, we use our phenotypic analysis method to compare and contrast the behaviors of teams evolved without communication to those evolved with communication in order to ascertain whether the evolved behaviors are different.

A. Capturing Behavioral Information

The first step in our phenotypic analysis approach is to specify a method for formally capturing each team agent's behavioral information from the gaming environment. To accomplish this, 1000 games are simulated for each team in which periodic snapshots of the team agents' game state are taken. These snapshots are taken every n game updates until either the game has ended or the team agent has been killed by the enemy. As this work is not concerned with the temporal analysis of agent behavior, all snapshots are accumulated into a single record describing the agent's behavior over the entire game. A sample record can be seen in Table I.

Each record holds information on the amount of damage inflicted on the enemy using each weapon, the average health of the agent, the average distances to other teammates and the enemy over the course of the game, the number of times the agent was the target of the enemy, the lifetime of the agent, the game length, and a Boolean value indicating whether the agent survived the game. One record is obtained per agent for each of the 1000 games played. These are later used to classify the team agents into specific agent types.

B. Classifying Agent Types

Agent types are identified that encapsulate the different behaviors that each agent can display in the game. Although these agent types are arbitrarily chosen from observation, they are deemed sufficient for the environments being examined in this paper. These types include: decoys, rocket attackers, shotgun attackers, railgun attackers, blaster attackers, evaders, health preservers, and cohesive agents. Each type has specific behavioral traits but types are not mutually exclusive. The agent types have associated degree of membership (DOM) algorithms that are used to calculate the DOM each agent has to every agent type by analyzing the captured behavioral information of that agent. The DOM algorithms used in this work have been revised and improved upon the ones used in previous research [32] to give more accurate classification of agent types. For a detailed explanation of the agent types and DOM algorithms used, see the Appendix.

The values returned from the DOM algorithms are used to calculate probability scores for each team agent that indicate how likely their behavior describes each type according to the following function:

$$\text{Final Probability} = \frac{1}{1 + R \times \exp^{((-\text{Value}) \times S)}}$$

where R and S are constant values and Value is the original value returned from a DOM calculation.

The probabilities of each agent being a particular type are calculated for each of the 1000 games and the results are averaged to obtain a vector of probabilities describing an agent's behavioral characteristics. This is carried out for each team agent resulting in five probability vectors that, when combined, describe the behavior of the team as a whole.

C. Comparing Squad Behaviors

When these probability scores have been calculated for every team agent, they are combined into a single team vector describing the overall team's behavior. In order to model different levels of membership to different role types, we subdivide the interval $[0, 1]$ into five intervals of equal size, where the first interval $[0.0, 0.2]$ represents low membership to that particular class and $[0.8, 1.0]$ represents a high level of membership to the particular class. We aggregate the scores indicating membership to these classes for each individual member to give an overall team vector. This team vector is represented by a four-tuple indicating the count of the number of members in each of the four classes indicating membership (note that we interpret the lowest range as nonmembership to the class and hence the count of membership to this class is omitted in our representation). Hence, when all agents have been added to the team vector, each class corresponds to the number of agents with weak, fair, strong, and very strong memberships to that type on the team. This technique for combining the agent vectors into a single team vector gives us a more accurate representation of the squad's behavior than the threshold method used in [32] by minimizing the amount of information lost. The team vectors are then put into a decision tree algorithm to be analyzed.

VI. EXPERIMENTAL SETUP

In order to explore the effects of communication on the evolution of squad behaviors, the environments, genetic program, and game parameters used for these experiments are identical to those used in previous work [10], in which teams were evolved without the use of explicit communication. In [10], teams have been evolved in eight shooter environments of varying difficulty. As there was no explicit communication, teams evolved to cooperate implicitly. It was found that the effectiveness of evolved teams decreases significantly as the environments become more difficult.

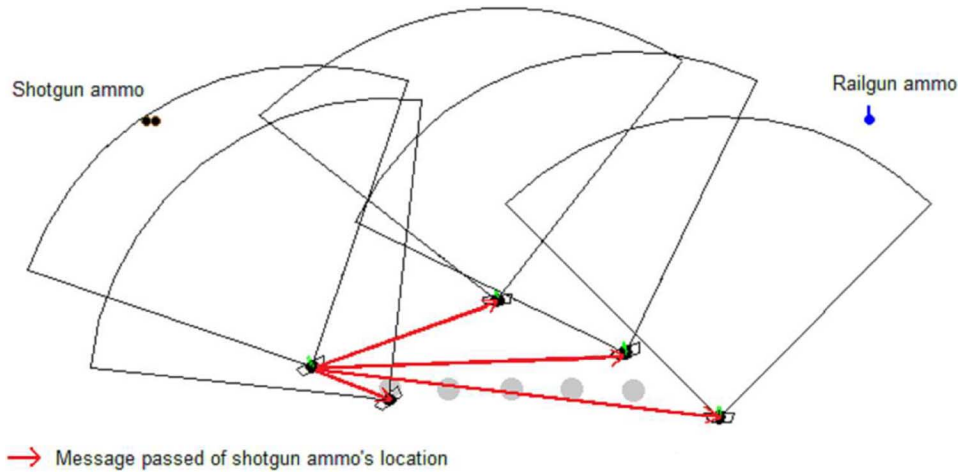

Fig. 6. Sharing perceived information with teammates.

The only difference between these experiments and those of previous research is that in these experiments NPCs are given the ability to share perceived information as the games are played. As game information is sensed by an NPC, it is broadcast by the NPC to each of its teammates in the form of messages. Each of the teammates then receives the message and stores the information in memory. Types of information that can be exchanged between teammates include the location of health and ammunition packs as well as the enemy's current position. A visualization of an agent informing teammates of the location of shotgun ammunition is shown in Fig. 6. In a commercial shooter game, audio cues could be used with this communication to enhance gameplay. For example, when the enemy is located, an NPC might say "enemy spotted at location X" as it passes the message to its teammates. We hypothesize that this sharing of game information between team members should allow the NPCs to perceive the environment as a team rather than individually and that this should result in more effective emergent squad behaviors. In the remainder of this paper, teams evolved in [10] will be referred to as noncommunicating teams and those evolved in this work that communicate perceived information will be referred to as communicating teams.

The difficulty of the environment is varied by altering the agents' visual perception capabilities. Experiments are set up for two fields of view (90° and 180°) and four viewing distances (50, 200, 350, and 500 pixels) so a total of eight experiments are conducted. The enemy viewing distance is scaled relative to the viewing distance of the team NPCs. As there are five team agents and only the one enemy agent, the collective viewing range of the team covers a much larger portion of the map than that of a single agent. Additionally, the human player in single-player shooter games, to which the enemy is likened, usually has a much longer viewing distance than that of the NPCs. For these reasons, it was decided to allow the enemy's viewing distance to be twice that of a team agent. Fig. 7 show the environments where the FOV is 90°.

Twenty separate evolutionary runs are performed in each of the eight environments. In each of the runs, 100 team chromosomes are evolved over 100 generations. Each team evaluation in each generation comprises 20 games. The best performing team from each of the runs is recorded.

As the enemy's behavior varies from game to game, due to the random biases in its desirability algorithms, each recorded team is tested more extensively using a larger number of games to obtain an accurate measure of its effectiveness. The effectiveness tests involve evaluating each recorded team's performance over 1000 games and recording the number of games won by the team out of the 1000. These results are then compared to previous results and statistical tests are performed to determine if communication provides a significant benefit to the evolution of effective squad behaviors.

A baseline of 100 generic team behaviors is also created and assessed in each environment to help put the effectiveness of the evolved teams in perspective. These generic teams consist of agents that use desirability algorithms to define their AI (similar to the enemy agent). In order to create 100 distinct behaviors, teams are split up into a group of three and a group of two. Each group then is given a fixed set of biases for each of its desirability algorithms where two of the biases are set high (1.5) and three set low (0.5). As there are ten ways of choosing two desirability algorithms for high biases from five possible desirability algorithms for each group and as there are two groups of agents on a team, 100 distinct teams can be created from all combinations of the two groups. Note that there are six desirability algorithms but the *Explore* desirability is set to a constant low number and is not given an associated bias.

As mentioned previously, a phenotypic analysis is performed in order to determine if the behaviors of teams evolved with communication are different than those evolved without communication. As mentioned in Section V-C, each evolved team's behavior is described by a vector of numbers. In order to compare the behaviors of communicating teams to the behaviors of noncommunicating teams, each team vector must be labeled according to its type, i.e., "communicating" or "noncommunicating." It is this type label that is used as the target variable of the decision tree so that the decision tree can attempt to distinguish between the behaviors of teams evolved with communi-

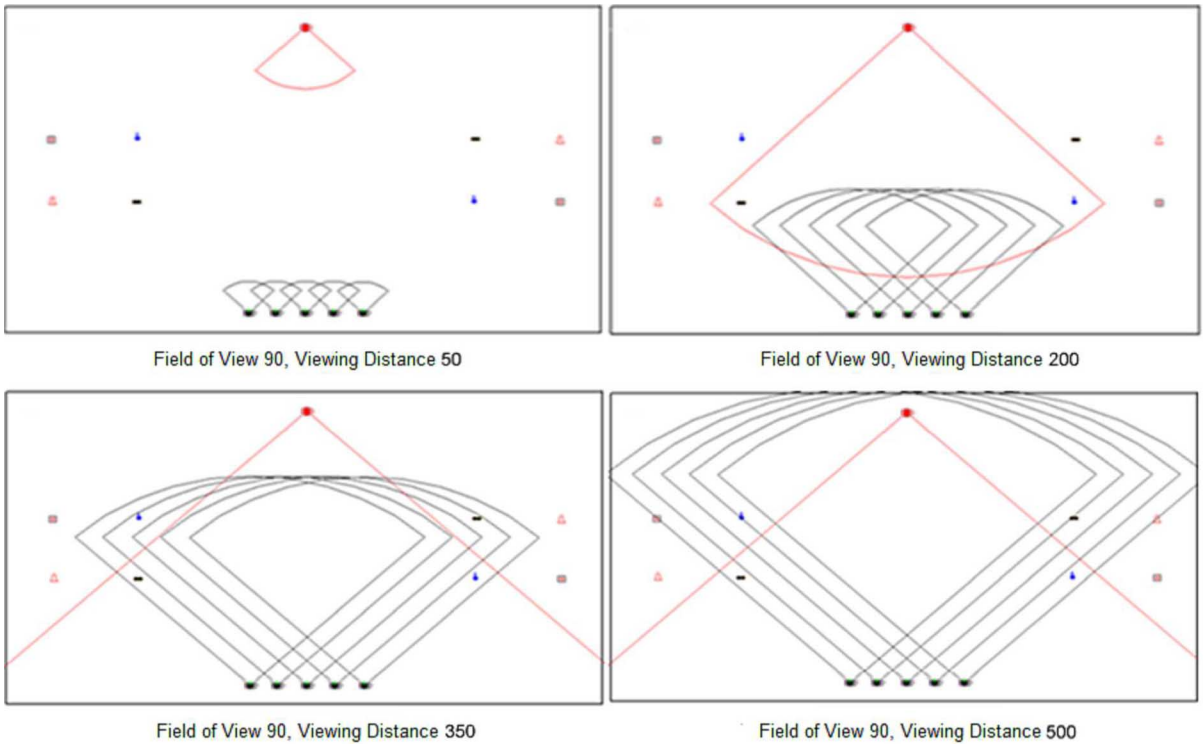

Fig. 7. Environments where FOV is 90°.

cation and those evolved without communication. We use tenfold cross validation for our decision tree analysis to give more accurate results as our data sets are relatively small (size 40). The higher the degree of misclassification within the tree, the more similar the behaviors are of the communicating and noncommunicating teams. If all behaviors are notably different, the decision tree should be able to classify them such that the leaf nodes of the tree have no misclassification, i.e., there are no teams evolved with perceptual communication that have similar behavioral traits to those evolved without perceptual communication. The path from the root node to a leaf should reveal the behavioral traits that are common to the teams grouped in that leaf.

As mentioned earlier, this paper is not concerned with performing a temporal analysis of the squad behaviors. Games cannot be split into an equal number of fixed length periods for comparison as game length is variable and agents have varying life spans[2] within each game so it is very difficult to compare teams on a temporal basis. If teams were to be compared on a temporal basis using our current approach, it would add greatly to the complexity of the team vectors making it more difficult to interpret and analyze the resulting decision tree.

VII. Results

Fig. 10 tracks the average and best fitness scores from two sample evolutionary runs to show how fitness changes over time (note that values closer to zero are better). From the graph, we can see that the average and best fitnesses from each run improve between the initial and final generations. The spikes in the graph denote a sharp change in fitness between generations, which result from variances in the enemy's behavior during each team evaluation as random biases are used for its desirability algorithms. These spikes tend to be less prominent in the later generations as the genetic program begins to converge on a good solution.

Fig. 8(a) and (b) shows the number of wins obtained by the most effective generic team and the most effective teams evolved with communication and without communication in the 90° and 180° FOV environments, respectively. The results show that the best evolved communicating and noncommunicating teams outperform the best generic team in the least difficult environments. In both sets of environments, the results show that communication causes an improvement in the most effective teams evolved in all environments except the least difficult environment. In the least difficult environments, the results for the best evolved communicating and noncommunicating teams are almost even, differing by only seven wins in the 90° FOV environment and four wins in the 180° FOV environment. We believe that communication does not benefit the teams in these environments as the individual NPCs can view the majority of the map by themselves and do not need their teammates to communicate the game information.

As the environments in Fig. 8(a) and (b) become increasingly more difficult, the percentage improvement in the effectiveness of the best teams evolved with communication over those without communication also increases. In general, perceptual communication seems to benefit the teams more as the viewing distance decreases. This is justifiable as team agents

[2]In some teams, all agents may survive the game, whereas in another team, two or three of the agents might get eliminated early in the game.

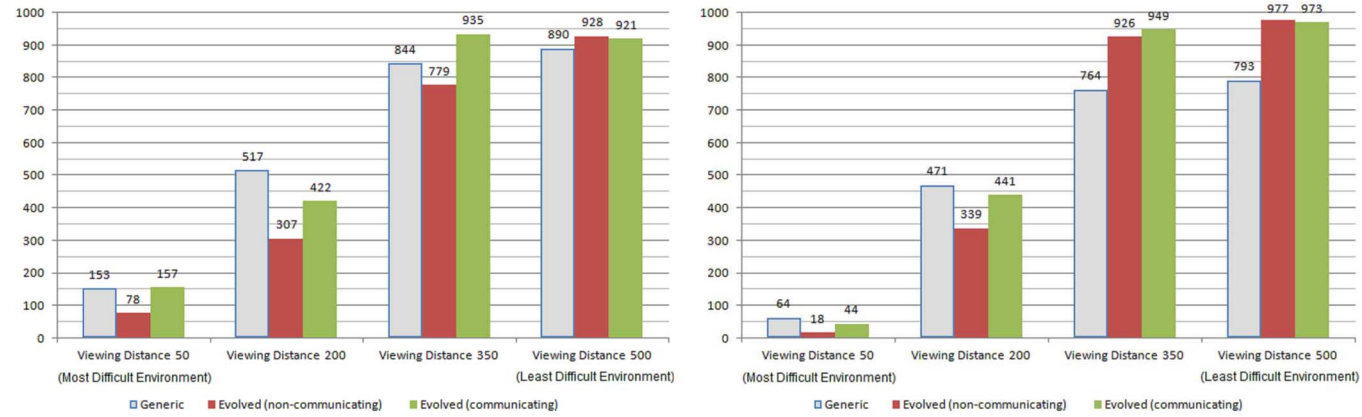

Fig. 8. Comparison of maximum wins. (a) FOV 90°. (b) FOV 180°.

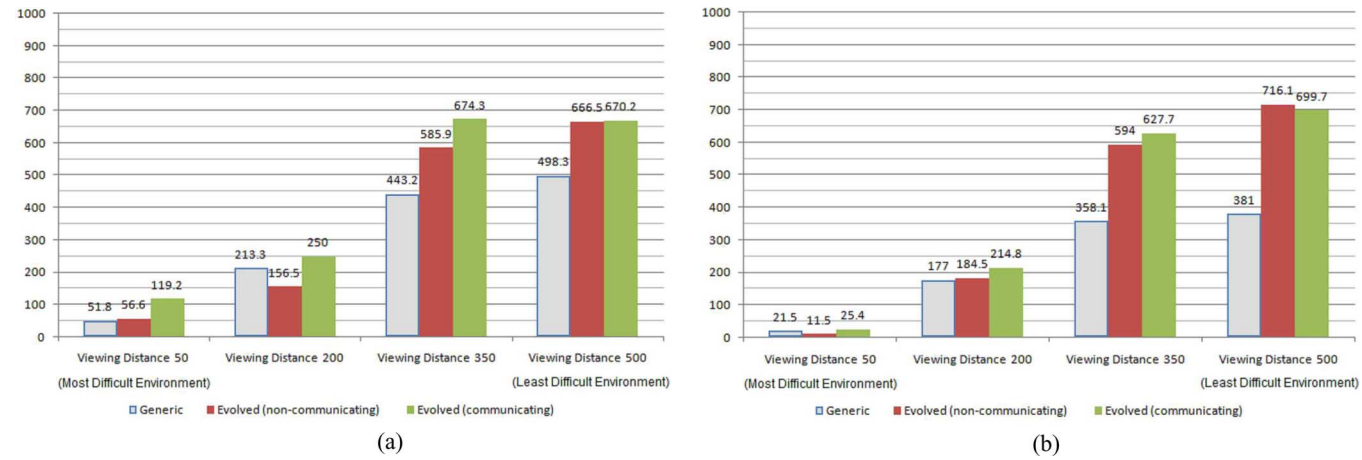

Fig. 9. Comparison of average wins. (a) FOV 90°. (b) FOV 180°.

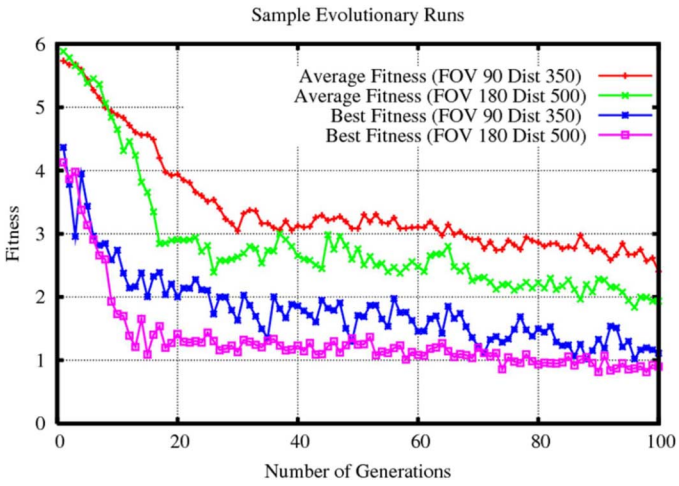

Fig. 10. Graph of the average and best team fitness from each generation.

with more restricted perceptual ranges would find it more difficult to locate specific game objects on their own, and thus should benefit more from sharing of game information.

Fig. 9(a) and (b) shows the average number of wins obtained by 100 generic teams, 20 evolved communicating teams, and 20 evolved noncommunicating teams in the 90° and 180° FOV environments, respectively. The results demonstrate that the average effectiveness of the evolved communicating teams is higher than the average effectiveness of the generic teams in all environments, showing that the GP gives a strong result. Similar to Fig. 8(a) and (b), excluding the least difficult environments, the results show an increase in the percentage improvement in average team effectiveness in those teams evolved with communication over those without communication as the environments become more difficult.

TABLE II
P-VALUES OF SIGNIFICANCE T-TESTS BETWEEN TEAMS EVOLVED WITH COMMUNICATION AND WITHOUT COMMUNICATION

	Field of view 90	Field of view 180
Viewing distance 50	< 0.01	< 0.01
Viewing distance 200	< 0.01	0.30
Viewing distance 350	0.05	0.61
Viewing distance 500	0.95	0.81

To test the significance of the results, T-tests have been performed between the evolved communicating and noncommunicating teams in each of the environments (see Table II). For a confidence interval of 95%, any comparison that records a p-value below 0.05 shows a statistically significant difference in the two samples. The results displayed in Fig. 9(a) show that

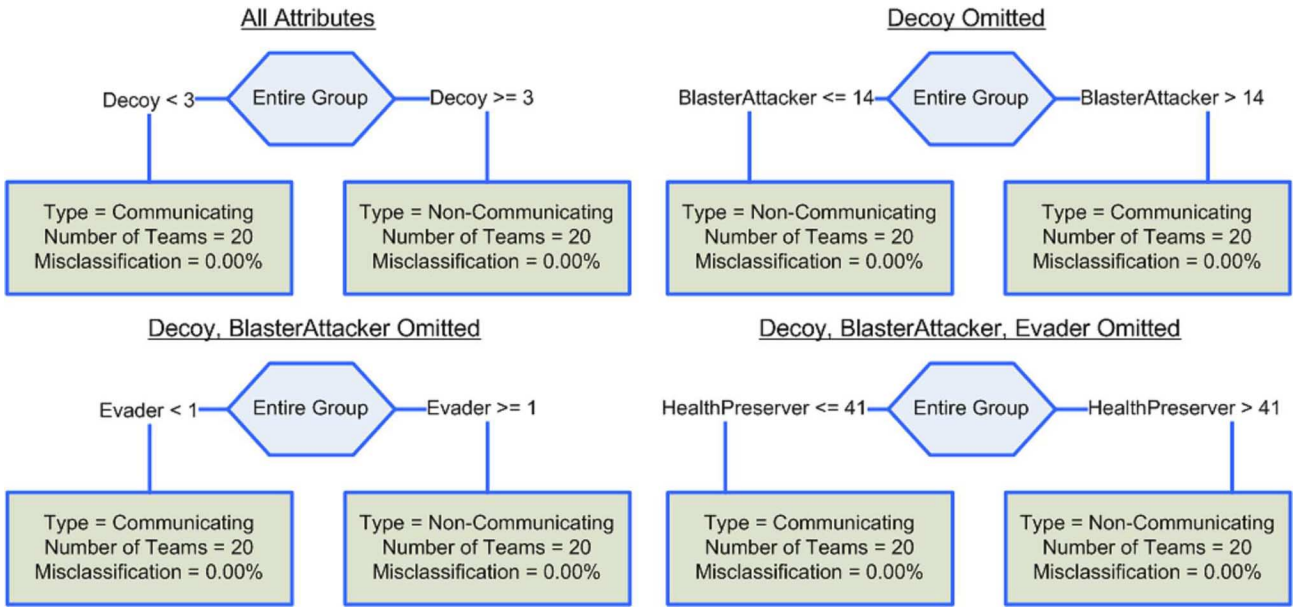

Fig. 11. Comparison of team behaviors (FOV 90°, viewing distance 50).

communication affords an improvement in the average effectiveness in all environments where the FOV is 90°. This improvement in performance is statistically significant for the 50- and 200-pixel environments (with p-values <0.01; see Table II) but is not significant for the 350- and 500-pixel environment (p-values of 0.05 and 0.95, respectively). In Fig. 9(b), the improvement in team effectiveness is only statistically significant in the 50-pixel environment. In addition, the use of communication causes a disimprovement in effectiveness in the 500-pixel environment but it is not significant.

Table II shows that in the 200-pixel environment the results are statistically different when the FOV is 90° (p-value <0.01) but not when the FOV is 180° (p-value of 0.30). Similarly, the low p-value (0.05) recorded in the 350-pixel environment where the FOV is 90° indicates that the results are very different but in the corresponding FOV 180° environment the results are similar (p-value of 0.61). This may because the enemy's FOV is also more restricted in the 90° FOV environments making it more difficult for the enemy to spot team agents attacking from the sides. This may also explain why the generic teams and some of the evolved teams tend to perform better in the FOV 90° environments than in their FOV 180° counterparts. Hence, communication between team members may provide the team with opportunity to attack more effectively. Additionally, the visual range of NPCs in shooter games is usually cone shaped, meaning their FOV is closer to 90° than 180°.

The results of our phenotypic analysis experiments show that there is a correlation between the difficulty of the environment and the difference in the behaviors of evolved communicating teams and evolved noncommunicating teams. Fig. 11 displays the resulting classification tree for teams evolved in the environment where FOV is 90° and viewing distance is 50. In this environment, the decision tree is able to distinguish all communicating teams from all noncommunicating teams in just one level of the tree. Moreover, Fig. 11 shows that the decision tree is capable of performing this classification within the first level of the decision tree using any one of four different agent type variables. This implies that the behaviors of the communicating teams are very different from the behaviors of the noncommunicating teams. More specifically, the communicating teams contain more agents with a higher probability of being blaster attackers, and a lower probability of being decoys, evaders, and health preservers than the noncommunicating teams. The improved performance of the communicating teams in this environment may be attributed to the greater number of higher probability blaster attackers on the team. The communicating teams have a more aggressive behavior than the noncommunicating teams, which is also reflected in their lessened ability to preserve their health effectively or evade the enemy. In addition, these teams contain fewer decoys which means that there are fewer ineffective,[3] self-sacrificing agents on these teams. From observation of the teams' behavior in this environment, it can be seen that the ability to communicate the position of the enemy enables the team members to attack the enemy simultaneously. Typically, the communicating teams evolved in this environment do not tend to collect ammunition for stronger weapons but instead approach and attack the enemy in unison when the enemy is spotted by a team member. This observed behavior reinforces the classification trees finding that the communicating teams contain agents with a higher probability of being blaster attackers and a lower probability of being decoys than the noncommunicating teams in this environment. The fact that agents approach the enemy as they attack also suggests that they do not attempt to evade the enemy and have little regard for their own health. A similar result is observed in the environment with corresponding viewing distance and FOV 180°.

In a contrasting example, the behaviors of both communicating and noncommunicating teams evolved in the environ-

[3]Decoys are only effective if the attacking agents on the team can take advantage of the distraction caused by the decoys.

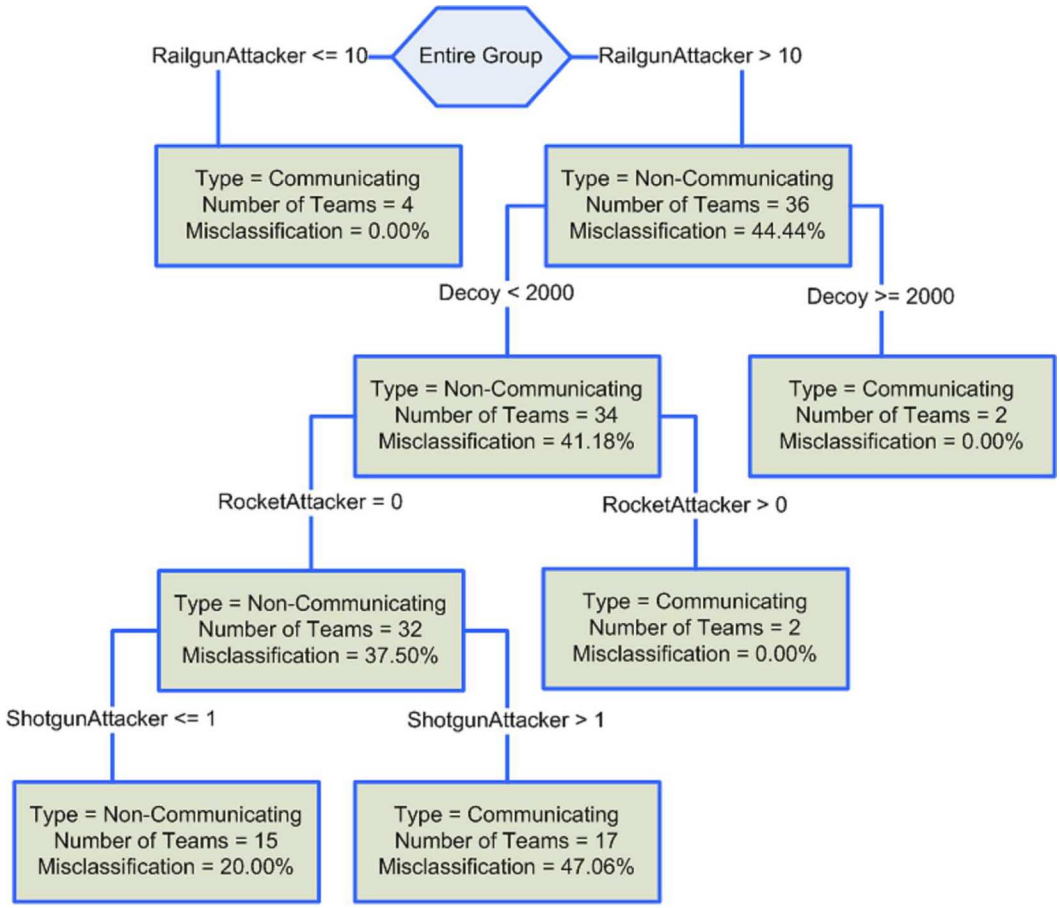

Fig. 12. Comparison of team behaviors (FOV 90°, viewing distance 500).

ment where agents have a FOV of 90° and viewing distance of 500 are found to be somewhat similar. The resulting classification tree for the teams evolved in this environment is shown in Fig. 12.

Only four communicating teams are classified in the first level of the trees and a further two teams are classified in each of the second and third levels of the tree. By the fourth level of the decision tree there are still 12 communicating teams that cannot be distinguished from noncommunicating teams based on their behavior as the leaf nodes at this level of the tree have misclassifications of 20% and 47%. This implies that the behaviors of communicating teams and noncommunicating teams contained at this level of the tree are similar. Both communicating and noncommunicating teams at this level of the tree contain at least one agent with a fair probability of being a railgun attacker, less than two agents with very strong probabilities of being decoy agents, and no rocket attackers. In this environment, only six teams are successfully classified within four levels of the decision tree, which would imply that the behaviors of communicating and noncommunicating teams are very similar. A similar result can be observed in the classification tree for the environment where FOV is 180° and viewing distance is 500. Typically, in these environments, the observed behavior of both communicating and noncommunicating teams is for one or two agents to distract the enemy (i.e., act as decoys), while the other team members collect ammunition for the stronger weapons before attacking the enemy simultaneously. Perceptual communication does not provide as much variation in team behavior in this environment as each individual agent's viewing range covers a much larger area and communication is not required to the same extent as in other environments.

The classification trees for the remaining more difficult environments (i.e., FOV 180° and viewing distance 50, FOV 90° and viewing distance 200, and FOV 180° and viewing distance 200) show that the decision tree is capable of distinguishing the behavior of all communicating teams from the behavior of all noncommunicating teams with no misclassification. In contrast, the classification trees for the less difficult environments, where FOV is 90° and viewing distance 350, FOV 180° and viewing distance 350, and FOV 180° and viewing distance 500, show some degree of misclassification, which implies that the decision trees are not capable of distinguishing between all the communicating and noncommunicating team behaviors.

VIII. Conclusion and Future Work

This paper explores the effects of shared perception on the evolution of squad behaviors for NPCs in shooter games. The results show that perceptual communication between team members enables an improvement in average team effectiveness in

all environments except the least difficult one. In the least difficult environment, individual NPCs can view the vast majority of the map by themselves and communication is not needed to inform them of key game information. In addition, the decrease in average team effectiveness when using communication is not statistically significant. In contrast, teams evolved in the more difficult environments, where NPC viewing ranges are most restricted, were shown to have a significant improvement in effectiveness when perceptual communication was used. The sharing of information by the team saves the NPCs having to explore the environment individually. On average, the evolved communicating teams are also shown to outperform hand-coded generic teams in all environments. Despite achieving a significant improvement in effectiveness in the most difficult environments, the evolved teams still only obtain win percentages averaging 11.9% and 2.5% in comparison to those achieved in the least difficult, which averaged 67% and 70% for FOVs 90° and 180°, respectively.

The current experiments show that as agents' visual fields become larger, the need for communication is reduced. This is because the only information being communicated is perceptual information. As an agent's own visual field becomes large, there is less of a need for teammates to inform them of the locations of game objects as they can more easily find the locations of objects themselves. This is reinforced by the finding that the behaviors of evolved communicating and noncommunicating teams are found to be more similar in environments where agents have larger visual fields. We hypothesize that information other than perceived game information which can be communicated among the agents, such as tactical commands, may be more important and may be unaffected by the broadening of visual fields.

In future, we wish to continue our research on the role of communication in facilitating teamwork by evolving behaviors on different types of map. Additionally, we wish to add communication nodes into the genetic program in an attempt to directly evolve effective tactical communication between the team members.

APPENDIX

Decoys are agents who deter enemy fire from their teammates. A decoy is classified by the average distance to the enemy over the agent's lifetime, the average distance to all of its allies over the agent's lifetime, and the number of times the enemy targets the agent as a fraction of the agent's lifetime. The number of times an agent is targeted is given a higher weight as it is a better indicator

$$\text{Decoy} = \left(\frac{\text{EnemyBoundary}}{\text{AvgDistEnemy}} \times \frac{\text{AvgDistAllies}}{\text{AllyBoundary}} \times 0.5\right) + \left(\frac{\text{EnemyTarget}}{\text{Lifetime}} \times 0.5\right).$$

Attackers are agents who consistently attack the enemy throughout the game. We define four different types of attacking agents, one for each weapon type. To find an attacker probability, the damage inflicted by the agent on the enemy over the agent's lifetime is calculated, taking into account the damage caused by a single round of ammunition for that weapon and the weapon's firing frequency

$$\text{RailAtkr} = \frac{\text{RailDmgInflicted}}{\text{RailDmg} \times \text{RailFireFreq} \times \text{Lifetime}}$$
$$\text{ShotAtkr} = \frac{\text{ShotDmgInflicted}}{\text{ShotDmg} \times \text{ShotFireFreq} \times \text{Lifetime}}$$
$$\text{RktAtkr} = \frac{\text{RktDmgInflicted}}{\text{RktDmg} \times \text{RktFireFreq} \times \text{Lifetime}}$$
$$\text{BlstAtkr} = \frac{\text{BlstDmgInflicted}}{\text{BlstDmg} \times \text{BlstFireFreq} \times \text{Lifetime}}.$$

Evaders are agents who try to keep their distance from the enemy agent to avoid getting killed. An evader's DOM score is based on average distance that the agent is away from the enemy over the lifetime of the agent. The larger this distance, the more likely it is that the agent is an evader. If the agent does not survive the game, it is penalized according to how short a life span it has

$$\text{Evader} = \frac{\text{AverageDistanceEnemy}}{\text{EvaderBoundary}} \times \exp^{-\left(\frac{C}{\text{Lifetime}}\right)}$$

where the second term is the penalty incurred if the agent is killed and C is a constant (set to 30).

Health Preservers are agents capable of surviving the game and they have a high average health level over their lifetime. If the agent survives the game its health preserver score is calculated as its average health over the game as a fraction of its maximum health. If the agent does not survive the game it is penalized according to how short a life span it has

$$\text{HealthPrsvr} = \frac{\text{AverageHealth}}{\text{MaximumHealth}} \times \exp^{-\left(\frac{C}{\text{Lifetime}}\right)}$$

where the second term is the penalty incurred if the agent is killed and C is a constant (set to 30).

Cohesive agents are agents who stay close to at least one of their teammates during the course of their lifetime. The average distance from the agent to its nearest teammate is checked to see if it is within some arbitrarily chosen distance

$$\text{CohesiveAgent} = \frac{\text{CohesiveBoundary}}{\text{AverageClosestAllyDistance.}}$$

ACKNOWLEDGMENT

The authors would like to thank the Irish Research Council for Science, Engineering and Technology for their assistance through the Embark initiative.

REFERENCES

[1] D. Doherty and C. O'Riordan, "Effects of communication on the evolution of squad behaviours," in *Proc. 4th Conf. Artif. Intell. Interactive Digit. Entertain.*, Palo Alto, CA, 2008, pp. 30–35.

[2] C. Thurau, C. Bauckhage, and G. Sagerer, "Imitation learning at all levels of game-AI," in *Proc. 5th Int. Conf. Comput. Games, Artif. Intell., Design Educat.*, 2004, pp. 402–408.

[3] T. Haynes and S. Sen, "Evolving behavioral strategies in predators and prey," in *Proc. Int. Joint Conf. Artif. Intell./Workshop Adapt. Learn. Multiagent Syst.*, S. Sen, Ed., Montreal, QC, Canada, 1995, pp. 32–37.

[4] S. Luke and L. Spector, "Evolving teamwork and coordination with genetic programming," in *Proc. 1st Annu. Conf. Genetic Programm.*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds., 1996, pp. 150–156.

[5] C. W. Reynolds, "An evolved, vision-based behavioral model of coordinated group motion," in *Proc. 2nd Int. Conf. From Animals to Animats 2: Simulat. Adapt. Behavior*, Cambridge, MA, 1993, pp. 384–392.
[6] M. LaLena, "Teamwork in genetic programming," M.S. thesis, School Comput. Sci. Technol., Rochester Inst. Technol., Rochester, NY, 1997.
[7] M. D. Richards, D. Whitley, J. R. Beveridge, T. Mytkowicz, D. Nguyen, and D. Rome, "Evolving cooperative strategies for UAV teams," in *Proc. 7th Annu. Conf. Genetic Evol. Comput.*, New York, 2005, pp. 1721–1728.
[8] S. Raik and B. Durnota, "The evolution of sporting strategies," in *Complex Systems: Mechanisms of Adaption*, R. J. Stonier and X. H. Yu, Eds. Amsterdam, The Netherlands: IOS Press, 1994, pp. 85–92.
[9] S. Luke, C. Hohn, J. Farris, G. Jackson, and J. Hendler, "Co-evolving soccer softbot team coordination with genetic programming," in *Proc. Int. Joint Conf. Artif. Intell./1st Int. Workshop RoboCup*, Nagoya, Japan, 1997, pp. 398–411.
[10] D. Doherty and C. O'Riordan, "Evolving team behaviours in environments of varying difficulty," *Artif. Intell. Rev.*, vol. 27, no. 4, pp. 223–244, 2007.
[11] J. Reynolds, "Tactical team AI using a command hierarchy," in *AI Game Programming Wisdom*. Hingham, MA: Charles River Media, 2002, pp. 260–271.
[12] B. J. Best and C. Lebiere, "Spatial plans, communication and teamwork in synthetic MOUT agents," in *Proc. 12th Conf. Behavior Represent. Model. Simulat.*, 2003.
[13] W. Van Der Sterren, "Squad tactics: Team AI and emergent maneuvers," in *AI Game Programming Wisdom*. Hingham, MA: Charles River Media, 2002, pp. 233–246.
[14] W. Van Der Sterren, "Squad tactics: Planned maneuvers," in *AI Game Programming Wisdom*. Hingham, MA: Charles River Media, 2002, pp. 247–259.
[15] A. J. Champandard, *AI Game Development: Synthetic Creatures With Learning and Reactive Behaviors*. Berkeley, CA: New Riders, 2004.
[16] M. Ponsen, "Improving adaptive game-AI with evolutionary learning," M.S. thesis, Faculty Media Knowl. Eng., Delft Univ. Technol., Delft, The Netherlands, 2004.
[17] N. Cole, S. Louis, and C. Miles, "Using a genetic algorithm to tune first-person shooter bots," in *Proc. Congr. Evol. Comput.*, 2004, vol. 1, pp. 139–145.
[18] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.
[19] P. J. Angeline and J. B. Pollack, "Competitive environments evolve better solutions for complex tasks," in *Proc. 5th Int. Conf. Genetic Algorithms*, San Francisco, CA, 1993, pp. 264–270.
[20] E. V. Siegel and A. D. Chaffee, "Genetically optimizing the speed of programs evolved to play tetris," *Adv. Genetic Programm.*, vol. 2, pp. 279–298, 1996.
[21] T. Ehlis, "Application of genetic programming to the "snake game"," *Gamedev. Net*, vol. 1, no. 175, 2000.
[22] K. O. Stanley, B. D. Bryant, and R. Miikkulainen, "Evolving neural network agents in the nero video game," in *Proc. IEEE Symp. Comput. Intell. Games*, 2005, pp. 182–189.
[23] S. Bakkes, P. Spronck, and E. O. Postma, "Team: The team-oriented evolutionary adaptability mechanism," in *Proceedings of the Third International Conference on Entertainment Computing (ICEC 2004)*, ser. Lecture Notes in Computer Science, M. Rauterberg, Ed. Berlin, Germany: Springer-Verlag, 2004, vol. 3166, pp. 273–282.
[24] M. Smith, S. Lee-Urban, and H. M. noz Avila, "Retaliate: Learning winning policies in first-person shooter games," in *Proc. 3rd Int. Conf. Artif. Intell. Interactive Dig. Entertain.*, J. Schaeffer and M. Mateas, Eds., 2007, pp. 1801–1806.
[25] S. Lee-Urban, M. Smith, and H. M. noz Avila, "Learning winning policies in team-based first-person shooter games," in *AI Game Programming Wisdom 4*. Hingham, MA: Charles River Media, 2008, pp. 607–616.
[26] T. Haynes, R. Wainwright, S. Sen, and D. Schoenefeld, "Strongly typed genetic programming in evolving cooperation strategies," in *Proc. 6th Int. Conf. Genetic Algorithms*, L. Eshelman, Ed., Pittsburgh, PA, 1995, pp. 271–278.
[27] D. J. Montana, "Strongly typed genetic programming," *Evol. Comput.*, vol. 3, no. 2, pp. 199–230, 1995.
[28] D. Chakraborty and S. Sen, "Computing effective communication policies in multiagent systems," in *Proc. 6th Int. Joint Conf. Autonom. Agents Multi-Agent Syst.*, New York, 2007, pp. 1–3.
[29] M. Barlow, M. Luck, E. Lewis, M. Ford, and R. Cox, "Factors in team performance in a virtual squad environment," in *Proc. Simulat. Technol. Training Conf.*, 2004, pp. 94–99.
[30] M. Buckland, *Programming Game AI by Example*. Plano, TX: Wordware, 2005.
[31] T. Haynes, S. Sen, D. Schoenefeld, and R. Wainwright, "Evolving a team," in *Working Notes: AAAI Symp. Genetic Programm.*, E. V. Siegel and J. R. Koza, Eds., Cambridge, MA, 1995, pp. 23–30.
[32] D. Doherty and C. O'Riordan, "A phenotypic analysis of GP—Evolved team behaviours," in *Proc. 9th Annu. Conf. Genetic Evol. Comput.*, New York, 2007, pp. 1951–1958.

Darren Doherty received the 1.1 B.Sc. (honors) degree in information technology from the National University of Ireland (NUI), Galway, Ireland, in 2005, where he is currently working towards the Ph.D. degree in game AI and evolutionary computation at the Department of Information Technology.

His main research interests include: evolutionary computation, AI, and computer game development.

Mr. Doherty is currently a member of the NUI Computational Intelligence Research Group (CIRG).

Colm O'Riordan received the B.Sc. degree (with honors) in computer science and the M.Sc. degree in computer science from University College Cork, Ireland.

Currently, he lectures in the Department of Information Technology, National University of Ireland (NUI), Galway, Ireland. His research interests are in the fields of agent-based systems, artificial life, evolutionary computation, and information retrieval. His current research focuses on cooperation and coordination in artificial life societies and multiagent systems.