Evolving Coordination for Real-Time Strategy Games

David Keaveney and Colm O'Riordan

***Abstract*—The aim of this work is to show that evolutionary computation techniques (genetic programming in this case) can be used to evolve coordination in real-time strategy games. An abstract real-time strategy game is used for our experiments, similar to a board game but with many of the properties that define real-time strategy games. We develop an automated player that uses a progressive refinement planning technique when determining its next immediate turn in our abstract real-time strategy game. We describe two types of coordination which we believe are important in the game and then define measurements for both. We perform twenty coevolutionary runs for our automated player and then analyze the history of each run with respect to the success of the solutions found and their level of coordination. We wish to show that as the evolutionary process progresses both the quality and the level of coordination in the solutions found increases.**

***Index Terms*—Coordination, genetic programming, real-time strategy.**

I. Introduction

REAL-TIME STRATEGY (RTS) games are military simulations where several players or teams attempt to destroy each others' units and infrastructure in order to be the last surviving player/team. The players compete for resources distributed across the map so that they can build armies capable of defeating their opponents. Often these games involve researching different technologies and building a base composed of many structures. These buildings are usually either for defensive purposes (e.g., a gun turret), to boost the player's economy (e.g., an ore refinery), to enable certain research options for the player to pursue or allow the construction of units helpful to the player in some way. Games such as "Starcraft," "Warcraft," "Command and Conquer," and "Age of Empires" are very popular RTS game series.

Real-time strategy games are a perfect example of complex multiagent systems. Classical games such as Chess or Go still prove to be extremely difficult problems to solve. However, the branching factor of such games, while large, still allows for forward reasoning (searching forward from the current game state to possible future game states to a reasonable depth of a few turns). Real-time strategy games can easily contain more units than these classical games have game pieces and require that all units of all players are moved in parallel. Therefore, the branching factor of such games is far larger than most classical board games and makes forward based reasoning approaches less helpful for planning. Real-time strategy games often adopt a goal-directed reasoning approach (or backward reasoning). This is where units, or groups of units (e.g., squads), have goals and then derive their actions backwards from these goals. This allows a player's behavior to be solved in a distributed fashion, where individual units can be modeled as agents. These agents can perceive their environment, communicate with other agents and each have their own beliefs and goals.

Problem solving can be handled at varying levels of abstraction, ranging from strategic to unit level. While automated players in RTS games today often exhibit good/acceptable tactical behavior, we feel they lack the ability for good strategic planning. Planning in RTS games can be distributed across different levels of abstraction through the use of a hierarchal task network (HTN) [1]. First, the problem is solved at the strategic level using high-level actions. After solving the problem at this level, the high-level actions that compose the solution can then each be reduced to lower level solutions, with the potential of producing one or more lower level actions/tasks. These in turn are decomposed into lower level partial plans until the lowest level is reached, where actions in the plan are equivalent to actions in the game.

Qualitative spatial reasoning (QSR) techniques can be used to reduce complex spatial states such as that seen in RTS games into abstract representations of that space [10]. Such abstraction methods are necessary for high-level planning. These spatial representations would look similar to how space is represented in many turn based strategy board games. We feel, that there is much completed research in the area of board games such as Risk and Diplomacy that can be brought to work on RTS games. However, most board games have different properties to RTS games. Most RTS games have imperfect information about the spatial state of the game (this is known as the fog of war) which is a property unknown to board games where the spatial state of the game is known to all players at all times. Also because RTS games take place in real-time, the success or failure of a player's actions can depend entirely on the simultaneously performed actions of the other players in the game (i.e., no player can be sure what every other player is about to do). This is different to most board games where each player takes individual turns and has perfect information of the game. Diplomacy is one of the few board games to model this type of imperfect information as players act in parallel. Our test-bed game for the work shown in this paper is called "Bellus Bellum Gratia" (BBG) [15] and serves as an abstract RTS game. It is similar in ways to a board game and includes imperfect information properties like those present in RTS games through the use of fog of war and parallel action phases.

In this paper, strategies are coevolved for this abstract real-time strategy game. This abstract RTS game is used primarily

Manuscript received June 29, 2009; revised December 21, 2009 and March 25, 2010; accepted December 13, 2010. Date of publication April 25, 2011; date of current version June 15, 2011.

The authors are with the Department of Information Technology, National University of Ireland, Galway, Ireland (e-mail: d.keaveney1@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.2011.2146783

because we only wish to learn coordination at this level of planning. It also reduces the computational expense associated with evolutionary computation techniques and renders results more amenable to analysis. These evolved strategies utilize a multiagent system framework and a progressive refinement planning technique that allow solutions to coordinate move and attack actions. We describe two types of coordination we feel are important in the game and define measurements for both. We intend to show that evolutionary computation applied in this way can be used to learn/evolve coordination.

Section II briefly discusses evolutionary computation and multiagent systems, that are used in our work. Other research in the area of evolutionary computation with respect to spatial decision making and multiagent system approaches to strategic board-games are discussed in Section III. The general rules of the abstract game used as a test-bed in this work are outlined in Section IV. Section V details the multiagent system automated player and the progressive refinement planning technique it uses. It also explains how the player's behavior is determined based on three functions. Section VI lists the functions and terminals (properties of the game) used by the genetic programming approach when evolving these three functions. Section VII describes the two types of coordination we believe are important in the game and defines general measurements for both. Section VIII describes the experiments performed to analyze the changes in the levels of coordination throughout the evolutionary process. Sections IX and X detail the results of these experiments and discuss our conclusions. Finally, in Section XI, some possible future directions for our research are discussed.

II. Background

A. Evolutionary Computation & Genetic Programming

Evolutionary Computation (EC) describes computing techniques inspired by biological evolution and other naturally occurring operations. EC techniques have the potential to be very computationally expensive. This is especially true in the computer games domain where many iterations of a game may need to be played in order to evaluate the fitness of each potential solution. Also, EC cannot guarantee to find an optimal solution. For these reasons EC techniques are not often used in commercial game development. However, these techniques have been used effectively in board games such as Chess [12], Go [16], Risk [13], and Diplomacy [13] to real-time games such as RoboCup [2], [4] and first person shooter (FPS) games [5], [7]. The basic idea behind all of these techniques is that the environmental pressure acting on a population of individuals causes natural selection (i.e., survival of the fittest) which results in a rise in the population's overall fitness over time [8]. First, an initial population of random individuals is created. Then, given a fitness function (i.e., a function that describes the ability of an individual solution to perform a given task well), the entire population is assessed and a new population is created by combining individuals selected from the current generation. When selecting these individuals there is a bias towards selecting fitter solutions. These new solutions can be constructed using crossover operators, which take two parent individuals to create new novel children, and mutation operators, which act to disrupt the information held by a solution and encourage diversity.

Genetic programming (GP) is one such technique that seeks to optimize computer programs based on their fitness. An important property of GP is that it can search through large search-spaces efficiently. The genetic programming paradigm represents individuals (i.e., computer programs) as hierarchical compositions of functions and terminals appropriate to the particular problem domain [18]. Furthermore, because the GP representational language uses a computer program structure neither the size of a solution nor its shape needs to be constrained beforehand. This property of GP allows for a much broader search thereby increasing the potential for better solutions.

B. Coevolution

In nature, coevolution occurs when two or more interacting species affect each other's evolution. Usually, when evolution is described, a species is adapting to a fixed environment when in actuality it is often adapting to those other species also living in that environment. Diffuse coevolution is the term used to describe evolution between groups of interacting species.

The fitness of individuals is determined by how well they perform while interacting with others of their species and other species. Because of this relative measure of fitness, once fit solutions, may become unfit in subsequent generations due to even better solutions being found. This can result in a "biological arms race" where individuals are continuously adapting to a likewise adapting environment. "Coevolution has been proposed as a way to evolve a learner and learning environment simultaneously such that progress arises naturally with minimal inductive bias" [9].

In this work, all solutions interact with all other solutions during evolution. There is no species distinction made between individuals and crossover can occur between all solutions (i.e., we use single population coevolution).

C. Multiagent Systems

A multiagent system (MAS) is a collection of two or more interacting, intelligent agents. An agent is generally considered to be a computer system that is situated within an environment, that has the capacity for autonomy in achieving its objectives [11]. In this case, autonomy refers to an agent's ability to act without the need for outside intervention. An intelligent agent is an agent that perceives it's local environment and acts accordingly to pursue its goals. It also has some degree of social ability that allows it to interact with other agents.

The multiagent system framework used by our automated player utilizes a forward planning refinement method that uses backtracking. Forward (or progression) planning constructs plans in a bottom-up manner by selecting actions, often using a heuristic, and then adding this action to the partial plan [6]. While such planners are not guaranteed to find an optimal solution, they often find strong plans quickly which is important in a real-time environment.

III. Related Work

Johansson and Haard [13] have investigated multiagent system approaches to designing player agents for the games Risk (MARS bot) and Diplomacy (HaAI bot). These MAS designed players performed very well against other automated players designed for those games. Both approaches opted for an entity-oriented agent system where each agent represents a territory or unit in the game. An evaluation function serves to assess the value of territories in either game and the values of parameters used in this function wholly determine the automated player's behaviour.

Work by Miles and Louis [17] shows how evolutionary computation can be used in spatial decision making in RTS games. A genetic algorithm determines how various influence maps, each representing a particular property in the game, should be combined to make new higher level influence maps that are then used to answer certain spatial decisions (e.g., good defensive positions, safe and bountiful resource positions). Their coevolved solutions were used directly as operational controllers rather than for strategic planning and produced behaviors that were as effective as their hand-coded strategies only more robust.

Evolutionary computation techniques have been used to evolve coordinated strategies in multiagent systems such as that seen in robot soccer teams. Work by Coelho *et al.* [4] investigate the evolution of coordination strategies in robot soccer teams. Each player (or agent) on a team is assigned to perform in a particular area on the playing field. Each of these team positions select their strategies for a unique population. They show, through coevolution, that by selecting players in such a way as to perform better with the already present team members, teams that act on complimentary areas of the pitch are evolved and are capable of defeating a team using a fixed formation.

EC has also been used to improve the behavior of automated players in first person shooter (FPS) games. Research by Cole *et al.* [5] has shown how genetic algorithms can be used to tune a FPS automated player's parameters to outperform parameters tuned by humans with expert knowledge. Work done by Bakkes *et al.* [3] describes a team-orientated adaptive mechanism (called "TEAM") which evolves team tactics offline, that outperforms previously seen static team AIs in the game *Quake 3*. Doherty *et al.* [7] investigate specifically how genetic programming can be used to evolve robust team behaviors in FPS games.

IV. A Brief Description of the Abstract Real-Time Strategy Game "BBG"

The game "BBG" can be represented by an undirected graph (like many board-games) (see Fig. 1 for the game map/graph used in this paper's experiments). Vertices represent the territories on the board (i.e., the hexagons) and edges the borders between territories. Each player starts the game controlling a territory known as their "homebase" (these are represented as the grey territories in Fig. 1). At the start of the game each player controls one unit located at their "homebase" territory. All units in the game have the same attack strength (a strength of 1) but can be stacked in the same territory to sum their strength together.

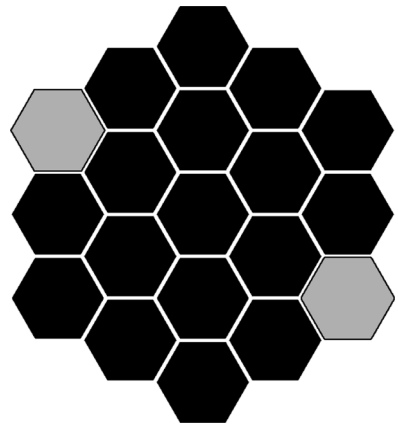

Fig. 1. Map used while evolving solutions.

Control of more territories increases a player's maximum allowed army size (an army is the set of all units belonging to a player) and if a player's current army size is less than their maximum army size then that player can receive new units at their homebase. Players can move to adjacent territories that are empty or occupied by friendly units and can attack adjacent territories that are occupied by enemy units. Occupying a territory for a whole turn, if already controlled by an enemy player, will neutralize the territory (remove the current controller) and if that territory is neutral controlled, will take control of that vertex.

When attacks occur, the strongest force wins the battle. The total attack strength is calculated as the sum of unit strengths involved in the attack on a specific territory. Attacks involving more than one territory attacking the same territory are given extra attack strength. This bonus is referred to as a "flanking bonus" and provides an additional twenty percent to the sum of unit strengths involved in the attack per extra attacking territory.

Should the defending force equal or outnumber the attackers, the attackers are simply repelled. However, should the attacking force be the strongest, losses are inflicted on the attacked territory.

Although the structure of the map is known, the dynamic information must be observed. Players can see all dynamic information (i.e., occupying units) in territories that are controlled, occupied, or adjacent to any occupied territory. A player wins the game if all the opponent player's units have been destroyed and it is no longer possible to make more (i.e., he does not control any territories or his homebase has been occupied).

V. Multiagent System Automated Player for BBG

A plan in the context of this research is the set of all "game orders" to be given during the current turn of the game. When creating a plan, the automated player creates a set of order agents for each territory in the game that can currently be attacked or to where units can be moved. An order agent represents a specific territory in the game. If this territory is empty or friendly occupied, owned units in this territory or adjacent territories can commit to this order. Should the territory be enemy occupied, only owned units in adjacent territories can commit to this order. Whenever an order agent successfully reaches the execution stage, all units committed to that order are issued with 'game orders' to move or attack from their current territory to

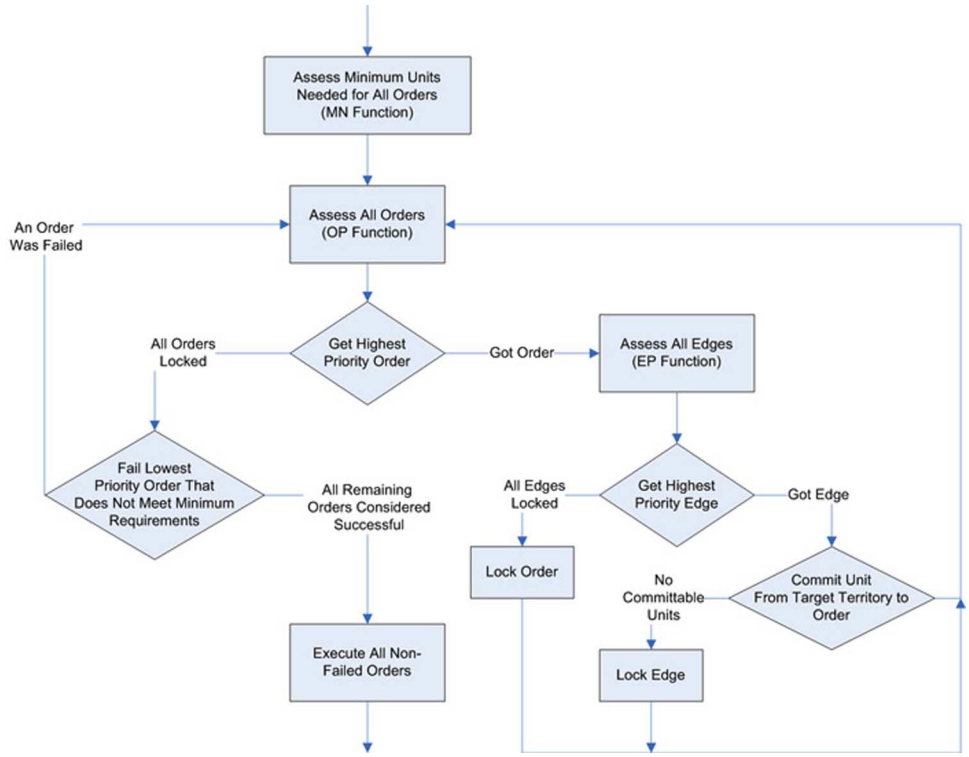

Fig. 2. Plan creation process.

the order's territory. That is, while an order agent represents all the units involved in moving to an area or attacking an area, a game order represents a subset of these units all of which are located at a specific territory with the intention of moving or attacking the order agent's territory.

A strategy in our game is determined solely by three behavioural functions: the "minimum needed" (MN) function, the "order priority" (OP) function and the "edge priority" (EP) function. The MN function determines the minimum required number of units necessary to go ahead with a specific order. During the incremental process of developing a plan, the OP function determines what orders should receive an extra unit and the EP function determines from which adjacent territory that unit is committed.

First, after all potential orders have been created, each is then assessed by the MN function and given a minimum unit requirement. The value remains constant throughout the plan creation stage. An order agent must meet this requirement for it to be considered successful and only successful order agents can be involved in creating the final plan.

Next begins a progressive refinement planning technique, whereby the current plan is created incrementally by committing one unit at a time to one of the available order agents. All the orders are evaluated using the OP function. This function's purpose is to select an order that will then attempt to commit one unit to itself. When this order has been found, another function, the EP function, assesses all the edges belonging to that order. These edges point towards all surrounding territories with units that can be committed and include a loop edge that points towards the order's own territory. The edge with the highest priority has the territory to which it points commit a unit to the highest priority order. This causes an incremental change in the current plan and then the same process of unit commitment occurs repeatedly until all the available orders have committed all the available surrounding units to themselves. Because the functions used to evaluate the orders and edges take certain plan-dependant parameters into account (such as the number of already committed units) the results of their evaluations can change as the plan changes (see Fig. 2 for a detailed guide to the plan creation process).

TABLE I
FUNCTION NODESET

Node	Description
$+$	Addition operator
$-$	Subtraction operator
$*$	Multiplication operator
$/$	Division operator
If()	Decision
$>$	Greater than operator
$<$	Less than operator
$=$	Equality operator

At this stage, all orders are once again evaluated using the OP function and sorted from worst to best. Beginning at the worst order and working towards the best, each order is checked to see if it has reached the minimum units required for its success (determined at the very start of plan formation by the MN function). The first unsuccessful order found is removed from the set of possible orders and has any units they had committed to themselves released of their commitments. Should this happen, this final order checking stage ends and the previous refinement planning stage begins again with the newly released units free to commit to other potential order agents and one fewer order agent

TABLE II
TERMINAL NODESET

Node Types	MN	OP	EP	Description
Enemy Strength	x	x		The number of enemy units occupying territory
Friendly Support		x	x	The number of friendly units occupying territory
Enemy Support	x	x		The number of enemy units in surrounding territories
Committed		x	x	The number of units in a territory committed to an order
Non Committed		x	x	The number of units in a territory not yet committed to an order
Friendly Connectivity	x	x	x	The number of surrounding friendly controlled territories
Enemy Connectivity	x	x		The number of surrounding enemy controlled territories
Unknown Connectivity	x	x	x	The number of surrounding friendly controlled territories
Minimum Needed		x	x	The minimum number of units needed for an order needs
Order Priority			x	The current priority of an order
Surrounding Friendly	x	x	x	The number of surrounding friendly occupied territories
Surrounding Enemy	x	x	x	The number of surrounding friendly occupied territories
Distance to Fr. Homebase		x	x	The number of territories on shortest path to friendly homebase
Distance to En. Homebase		x	x	The number of territories on shortest path to enemy homebase
Set of Constants	x	x	x	1, 2, 3, 5, 10, 25, 50, 75, 100 (Each are individual nodes)
Enemy Homebase	x	x	x	Is the territory the enemy homebase? (True/False)
Friendly Homebase	x	x	x	Is the territory the enemy homebase? (True/False)
Enemy Controlled	x	x	x	Is the territory the enemy controlled? (True/False)
Friendly Controlled	x	x	x	Is the territory the friendly controlled? (True/False)
Neutral Controlled	x	x	x	Is the territory the neutral controlled? (True/False)
Attack Order	x	x		Is the order an attack order? (True/False)

TABLE III
FITNESS FUNCTION

T_{end}	Number of turns the game lasted
T_{max}	Maximum number of turns per game
$Controlled$ $(0.0 - 1.0)$	Fraction of controlled territories over the total number of territories at the end of the game

TABLE IV
SPREAD COORDINATION MEASURE

O	Set of occupied territories
B_i	Set of enemy controlled or neutral controlled territories adjacent to territory i (includes territory i if enemy or neutral controlled)
c_i	# occupying units of territory i
A	$i \in A \mid \iff \big(i \in O\big) \wedge \big(\big\lvert B_i \big\rvert > 0\big) \wedge \big(c_i > 0\big)$
d_i	# separate move orders out of territory i into territories contained in set B_i

to consider. In this way, some unsuccessful order agents may get another chance to reach their minimum unit requirement.

Eventually, the automated player will be left with a set of considered successful orders that make up the plan for the current turn of our game.

VI. GENETIC PROGRAMMING

We use genetic programming (GP) to evolve strategies for our abstract RTS game. Three functions determine the behavior of our MAS automated player: the MN, OP and EP functions. Our genotype contains three different trees to evolve all three functions in parallel. Each of these trees has its own nodesets and rules which determines the possible structures they can take.

Table I shows all of the function nodes we use when evolving solutions. All three trees use these same function nodes. Table II shows all of the terminal nodes we use when constructing trees as solutions. Not all of these nodes are used by each of the three trees in our genotype. The table marks those nodes used by a specific function with an 'x'.

TABLE V
ATTACK COORDINATION MEASURE

A	Set of attacked territories
B_i	Set of surrounding friendly occupied territories of territory i
C_i	Set of territories involved in attack on territory i

A. Coevolution and Fitness Evaluation

Each member of the population plays a set number of games against other members of the population chosen at random. Each game gives a fitness. The overall fitness of an individual is the average of all fitnesses from all played games. The fitness per game is based on how quickly a solution wins or loses the game

$$\begin{aligned}\text{Fitness}_{\text{win}} &= T_{\text{end}} \\ \text{Fitness}_{\text{draw}} &= (1.2 * T_{\max}) - (\text{Controlled} * 0.2) \\ \text{Fitness}_{\text{loss}} &= (2.2 * T_{\max}) - T_{\text{end}}.\end{aligned}$$

The scalar values seen in the fitness evaluation function were chosen to allow for a small range of fitnesses that assess the end state of a game given the case of a draw.

Solution length penalization is also used. This is also known as parsimony pressure and encourages smaller solutions over larger ones.

B. Succeeding Generations and Reproduction Operators

When creating the next generation of individuals we use elitism to clone the fittest individual from the previous population directly into the new population. This ensures the survival of our fittest solution from generation to generation.

All individuals chosen for reproduction are selected using a tournament selection method. This method involves selecting a subset of individuals chosen from the evaluated population at random and then the fittest solution within this subset becomes the selected individual. The size of this subset is called the tournament size. It is a popular selection method as it provides an easily tuneable parameter (the tournament size) that controls how biased towards an individual's fitness it should be when

selecting an individual (i.e., the larger the tournament size, the less chance weaker individuals will be selected).

Other than crossover reproduction occurring, two types of mutation reproduction are also used. The three reproduction operators used when creating a new generation of individuals are now described.

1) Crossover Operator: The crossover operator used during the evolutionary process first selects two parent strategies. Next, one of the three subtrees present in all solutions is chosen (i.e., one of the three functions). Parent "A" then has a random node on that subtree selected. Another node is then chosen randomly on Parent "B" from the same subtree. If this node is not of the same node-set as that of the node chosen from parent "A" another node is randomly chosen from the subtree. When both selected nodes are of the same node-set, they are swapped with each other, including all descendant nodes. This produces two new trees which are the children of parents "A" and "B." Both of which are added to the new population.

2) Single Node Mutation Operator: This type of mutation operator acts on a single solution and selects a random node from any of the three subtrees. This node is then replaced with a different node from the same node-set that must have the same number of children as the selected node and have the same node-set restrictions for each of these children. It is possible that the first selected node is irreplaceable. In which case, no mutation occurs. This type of mutation is not very destructive to the genotype.

3) Sub-Tree Mutation Operator: This type of mutation operator acts on a single solution and selects a random node from any of the three subtrees. This node is then replaced with a new subtree, the root node of which must be of the same node-set. This type of mutation is more destructive to the genotype than the single node mutation operator described previously.

VII. Spatial Coordination Measures

Throughout the whole game, gathering resources is very important. This is done by taking control of territories on the map. Each territory gives a bonus to the number of units a player can support and at each turn a player receives a set number of additional units until that player's total unit support is reached. Therefore, the more territories a player controls on the map, the larger the army that player will receive which in turn should increase that player's chance of winning.

In order to spread quickly, it is important to coordinate the spreading of units. For example, given two units occupying an already controlled territory, each with the option of moving into one of two uncontrolled unoccupied territories ("territory a" and "territory b"), there are six possible plans that can occur: both could remain in their current territory, one could remain while the other moved to "territory a," one could remain while the other moved to "territory b," both could move to "territory a," both could move to "territory b," or, the quickest way to gain control of both territories, one could move to "territory a" while the other moved to "territory b." This type of coordination we call "spread coordination."

This type of coordination should dominate the opening game-play of successful strategies. Later in the game when both players meet, attacks can occur. Territories not yet controlled become more and more likely to be contested by the players. As attacks benefit from the number of separate territories involved in the attack (i.e., the total number of territories involved in the attack acts as a multiplier for all the units involved in the attack), we expect that units will be committed to an attack in a coordinated way. We expect to see attacks committing at least one unit from all the surrounding territories that could possibly commit a unit, as this will yield the greatest multiplier when calculating the total strength of the attack. This type of spatial coordination we call "attack coordination."

A. Spread Coordination Measure

Our global measure of spread coordination (C_S) is the average of all locally perceived spread coordination at territories contained in the set A. The set A contains all the territories that are occupied and have the potential to spread into surrounding enemy or neutral controlled territories (i.e., there are uncontrolled territories to move into and there is at least one unit occupying the territory that can move into these uncontrolled territories)

$$C_S = \begin{cases} \frac{\sum_{i \in A} (d_i / Min(|B_i|, c_i))}{|A|}, & \text{if } |A| > 0 \\ \text{not applicable}, & \text{if } |A| = 0. \end{cases}$$

This measure assesses how well a strategy has moved units into as many uncontrolled territories as is possible during a turn in our game.

A spread coordination of 1.0 denotes perfect spread coordination (i.e., the total number of uncontrolled territories possible to occupy were occupied). A spread coordination of 0.0 is the worst possible value and describes a situation where no uncontrolled territory is occupied (assuming that such territories are possible to occupy).

B. Attack Coordination Measure

Our global measure of attack coordination (C_A) is the average of all locally perceived attack coordination against the set of all attacked territories (i.e., set A)

$$C_A = \begin{cases} \frac{\sum_{i \in A} \left(\frac{|C_i|}{|B_i|} \right)}{|A|}, & \text{if } |A| > 0 \\ \text{not applicable}, & \text{if } |A| = 0. \end{cases}$$

This measure determines how well a strategy utilizes as many friendly occupied territories surrounding attacked territories in those attacks and therefore, how well it avails of the flanking bonus provided by extra territories involved in an attack.

The maximum value for attack coordination is 1.0 and describes a situation where all attacked enemy territories were attacked to some degree by all surrounding friendly occupied territories. Lower values of attack coordination suggest not all surrounding friendly occupied territories participated in the attacks. A value of zero is not possible, since in order to classify a territory as attacked at least one territory must be involved.

C. Coordination Measures: Discussion

Each of these coordination measures attempts to measure the total tactical level of coordination across the whole map for each

TABLE VI
GP SETTINGS

Parameter	Value
PopulationSize	500
NumberOfGenerations	500
CrossoverProbability	95
CreationProbability	5
TournamentSize	5
SingleNodeMutationProbability	30
SubTreeMutationProbability	2
NumGames	40
MaxTurnsPerGame	200
LengthPenalisationFactor	0.02

turn of the game. These measures are essentially the average of all locally perceived degrees of coordination (tactical coordination) and since these local views often overlap there is often some interference between these instances of locally measured coordination. For example, two units, each from different territories moving into the same unoccupied uncontrolled territory when only one unit was necessary. It is therefore important to note that there may exist more coordinated states that are undetectable using the above described measurements. This is further complicated by other issues such as enemies acting against the success of actions, requiring the commitment of more than one unit in order to be successful. In such situations, spreading too thin may not be advantageous. These measures also ignore any temporal element to a good spreading behavior such as the need to stop in a territory for a set time in order to take control of it and avail of the increased army size limit controlled territories provide.

VIII. EXPERIMENTS

The purpose of these experiments is to see whether our GP-evolved successful strategies that exhibit a high degree of coordinated behavior.

We evolve strategies for our abstract RTS game over 20 runs of evolution. The GP parameters for all these runs can be seen in Table VI.

During the course of evolution the best strategy (the strategy with the highest fitness) at each generation is saved for later analysis. A sample set of these strategies for each run then takes place in a round robin tournament involving 1000 games between all pairs of strategies (where strategies can play themselves). Each sample set contains the best strategies found at generations 1, 2, 3, 5, 10, 50, 100, 300, and 500.

The game is played on the same map during all runs of evolution and can be seen in Fig. 1. Each hexagon in Fig. 1 represents a territory of the map and the two shaded territories represent the homebases (starting locations) for both opposing players.

During these tournaments, we create a coordination profile and a control profile for each player–player pairing. A coordination profile records how coordinated a player is when spreading and attacking. A control profile records the average number of controlled territories at each turn in the game. A control profiles allows us to determine how successful a spreading behavior truly is.

During each game, at every turn, the coordinated spread and coordinated attack values (if applicable) are measured for each player. The average value of those measures in the profiles is

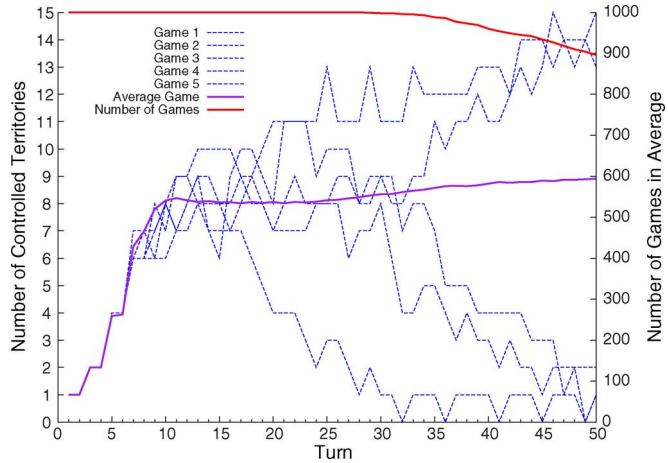

Fig. 3. Average control profile for the 5-5 player–player pairing.

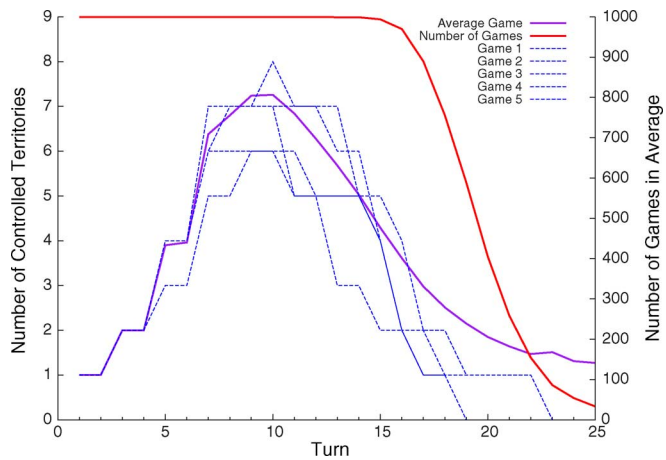

Fig. 4. Average control profile for the 5-500 player-player pairing.

then updated. It is important to note that these average values may only represent a subset of all the played games because as games finish they can no longer contribute to the average of later turns. Fig. 3 shows the control profile for generations 5 played against itself. It also shows five actual games that contributed to this average and the number of games that make each average at each turn.

Fig. 4 further demonstrates the previous point of how the average values of these measures may only represent a subset of all the played games, as games played against a much stronger opponent are more likely to end quickly. In this figure the number of games representing the average game quickly drops off between turns 15 and 20. Therefore, for this player-player pairing only the data graphed until turn 15 is representative of all games played by the pair.

These player-player profiles can then be averaged together for each player to get an overall coordination profile for that strategy. These averaged profiles are then compared to each other in order to evaluate how these measures of coordination change throughout the average game as the evolutionary process progresses. These generalized profiles show an average from a diminishing subset of games as turns progress (as seen in Fig. 4). The first fifteen turns for all profiles generally represent all played games, but then, as games begin to end, the trend more

TABLE VII
THE AVERAGE COORDINATION MEASURE

$\bar{C}_t = \frac{Total\_C_t}{\#Games_t}$	
$\bar{C}_t$	The average coordination measure at turn t
$Total\_C_t$	The sum total of all the coordination measures at turn t over all games
$\#Games_t$	The number of games that contributed to $Total\_C_t$

and more represents games that take longer to finish. Therefore these profiles are only general for the first fifteen or so turns. Table VII shows the formula used to obtain the average coordination measure at each turn in the game.

A. Coordination Measurement Analysis

Attack coordination is not applicable in the first few opening turns of the game. After the second turn in our game attacks becomes possible and our measures for spread coordination and attack coordination are no longer independent of each other. At this very early stage of the game, we expect to see our initially high spread coordination begin to drop as attacks are more and more likely to take from the spreading behavior.

From previous experience with evolving strategies for our game [14], aggressive solutions proved to be very successful, and therefore, we would expect to see our measure for attack coordination to be very high from as soon as it becomes applicable (after turn two or three) to the end of the game. As evolution progresses, we would also expect to see less interference between attack and spread coordination.

IX. RESULTS

After analysing the coordination profiles for all strategies within the sample set for each run, we determined that coordination does appear to evolve, in general, as expected. In analysing each run we considered three data sources. Our primary sources of data are the coordination profiles for each solution. Our secondary sources of data are the control profiles for each strategy (a record of the average number of controlled territories at any turn in the average game for each solution). And our tertiary source of data for each run is the tournament results. We categorized each evolutionary run into one of four categories: expected results, unexpected coordination profiles, unexpected control profiles and unexpected coordination profiles and control profiles. A smaller subset of strategies for each of our original sample set is shown in the following results for the purpose of increased clarity.

A. Expected Results

As already stated, in general, most evolutionary runs (i.e., 16 of 20) fall into this category. An analysis of one of these typical runs is discussed in this paper and is referred to as run A.

As expected, strategies in run A evolved towards a high attack coordination (see Fig. 5). The initially low value for attack coordination is due to the fact that attacks are not possible during the first two moves because players are situated a distance of three

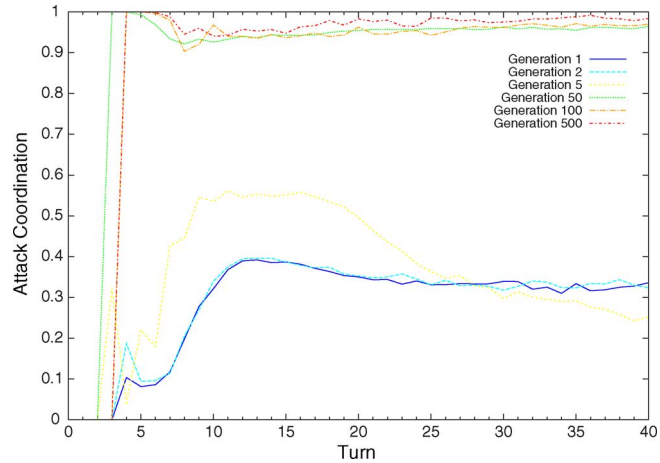

Fig. 5. Attack coordination for run A.

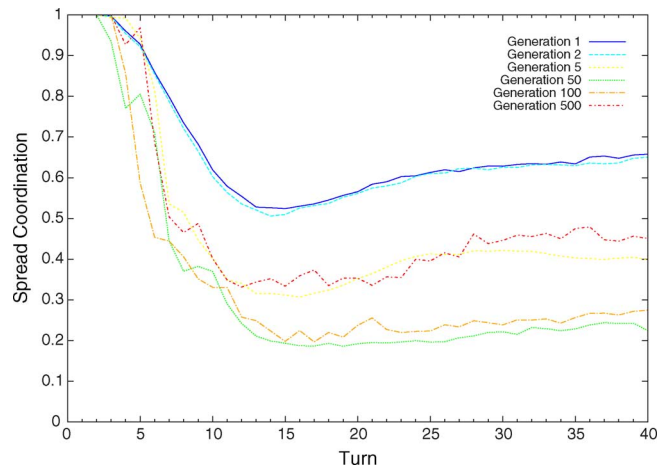

Fig. 6. Spread coordination for run A.

away from each other at the beginning of all games. Attack coordination continuously increases throughout this run until generation 100.

Spread coordination for all solutions is extremely high during the first opening turns of all games and then falls off as attack coordination comes into play. Initially spread coordination in this run is high but then begins to drop off as the attack coordination becomes more prevalent (see Fig. 6). However, during generation 100 and 500 while there is little change in attack coordination, the spread coordination begins to rise again. Solutions that attack well seem to be more important than solutions that spread well and when good attack coordination is evolved only then does better spread coordination evolve.

As can be seen in Fig. 7, the number of territories taken under control during the early game increases as evolution progresses. Controlling territories is important to a good strategy as they are the equivalent to resources in the game and allow a player to produce more units proportional to the territories controlled.

Table VIII shows the round robin tournament results for run A. Each column details the number of wins a strategy had against other strategies. The diagonal details the number of wins a strategy has against itself. This only serves to show how many games came to a definite end (won or lost) and how many

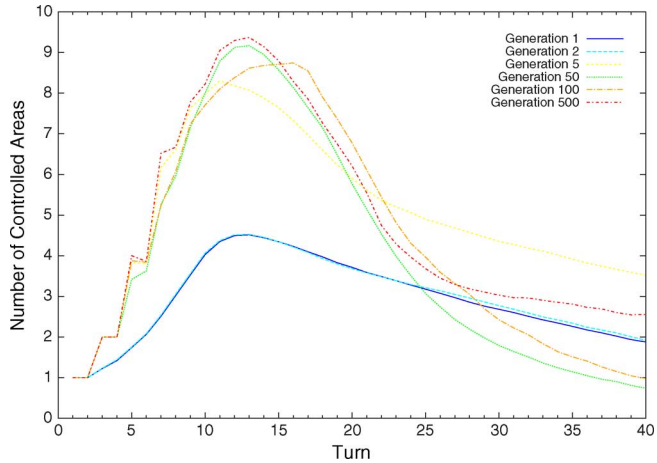

Fig. 7. Control profile for run A.

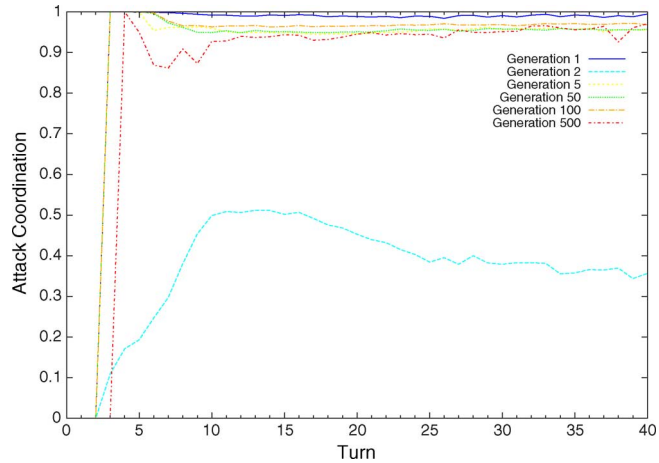

Fig. 8. Attack coordination for run B.

TABLE VIII
TOURNAMENT RESULTS FOR RUN A

	1	2	5	50	100	500
1	1000	497	523	993	994	996
2	503	1000	500	990	993	998
5	271	270	483	999	1000	998
50	6	10	0	964	641	987
100	5	6	0	358	716	565
500	0	0	0	13	430	358

were draws. These results for run A show strategies 1 and 2 to have performed very similarly. This is because they are the same strategy (the best strategy found in the first generation was still the best in the second generation). Later evolved strategies perform better or similarly in the tournament against all other strategies compared to earlier ones (i.e., the best strategy at generation 500 is a dominant strategy against the other four strategies).

The best strategy found at generation 5 draws often against itself and previously evolved solutions. Generation 500 appears to be the most successful strategy against all other strategies, while forcing a draw with itself most of the time (742 draws out of 1000 games).

B. Unexpected Coordination Profiles

Only two of the twenty runs of evolution produced data with unexpected changes in the coordination profiles. The results for both were similar. Therefore, we discuss the results of only one of these runs and refer to it as run B.

In run B, the best evolved strategy in the first generation has the best attack coordination than any other strategy evolved thereafter (see Fig. 8). This quickly falls in generation 2 only to rise back nearly as high in generation 5. Generation 100 improves a little on its attack coordination and then in generation 500 we notice that it drops again to slightly below that seen at generation 5. This differs from run A where the attack coordination seemed to increase as the evolutionary process progressed.

The spread coordination profiles for run B (see Fig. 9) drop progressively from generation 1 to 5 and then shows improvement from generation 100 to 500, similar to that seen in run A.

The control profiles for run B (see Fig. 10) clearly show a dramatic increase from generation 1 and 2 to 5. Given that the

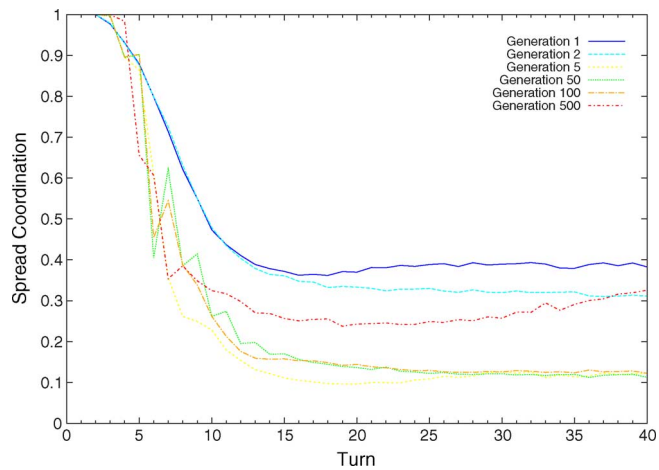

Fig. 9. Spread coordination for run B.

spread coordination for these first two strategies is quite high, it is clear that a strategy with high spread coordination is not the same as one that takes control of territories quickly. A strategy could spread very well but not remain in a territory long enough to take control of it. Therefore, it would seem that high spread coordination does not always imply a smart spreading strategy.

The tournament results for run B (see Table IX) show that generation 1 performs only slightly better than the best from generation 2 (understandable given how diverse these initial populations are). Generations 5 and onward once again show improvement as evolution progresses as seen in run A.

C. Unexpected Control Profiles

One of the twenty runs of evolution produced unexpected changes in the control profiles as the evolutionary process progressed. This run we refer to as run C.

The attack coordination profiles for run C (see Fig. 11) develop somewhat similarly to run B. They start low but drop much further in generation 5. However, the much later strategies found at generation 100 and 500 evolve a very high attack coordination.

The spread coordination profiles for run C (see Fig. 12) show an increase in generation 5 which would explain the

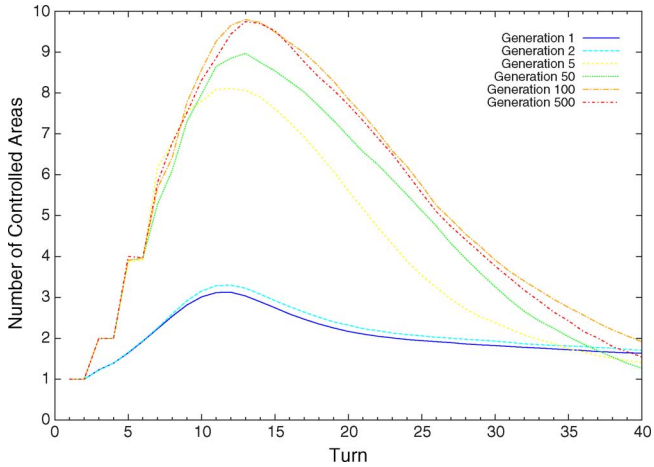

Fig. 10. Control profile for run B.

TABLE IX
TOURNAMENT RESULTS FOR RUN B

	1	2	5	50	100	500
1	774	364	998	1000	999	1000
2	419	777	997	999	1000	1000
5	0	1	879	999	1000	999
50	0	1	0	996	801	966
100	1	0	0	199	984	879
500	0	0	0	34	121	522

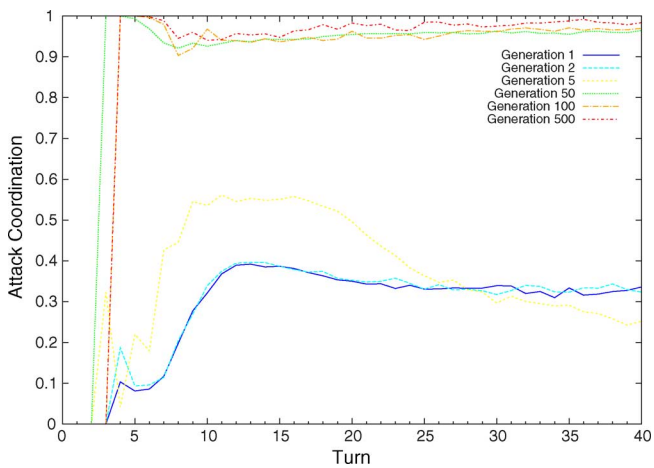

Fig. 11. Attack coordination for run C.

corresponding fall in attack coordination. A higher spread coordination than was seen in the previous generations produced a better strategy at the expense of some attack coordination. This then falls again in generation 100 as the attack coordination increases substantially, but rises again at generation 500 as the attack coordination is very high.

The control profiles for run C (see Fig. 13) show a continual decline in controlled territories during the early stage of the game most noticeably, the drop between generation 100 and 500. This seems very counterintuitive since as already mentioned the number of controlled territories is directly proportional to the number of units available to a player. The rise in attack coordination from generation 100 to 500 appears to be responsible for the drop in spread coordination and likewise the

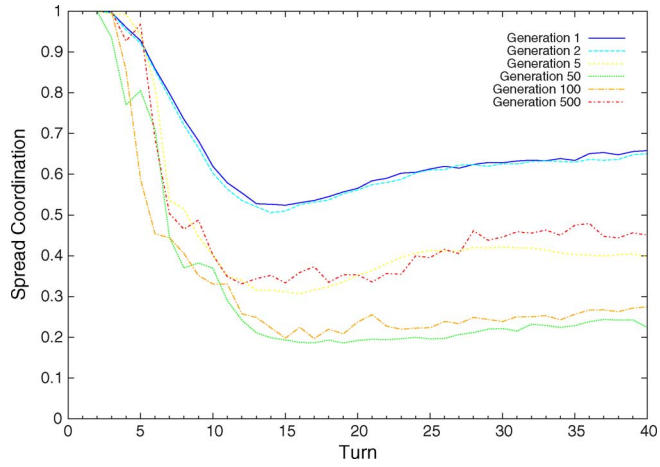

Fig. 12. Spread coordination for run C.

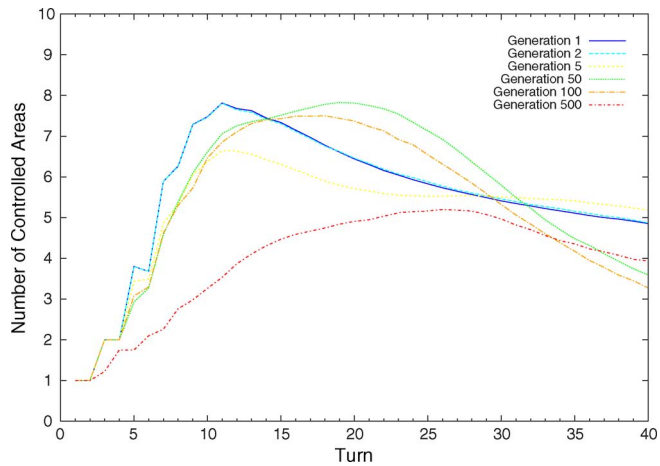

Fig. 13. Control profile for run C.

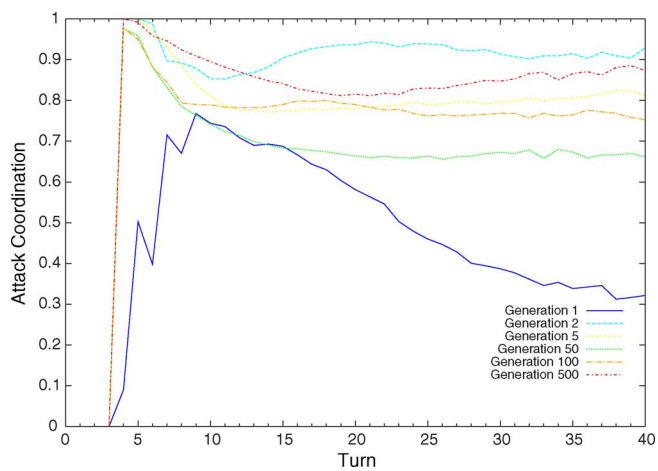

Fig. 14. Attack coordination for run D.

drop in the controlled territories. This highlights the importance of an aggressive strategy.

The tournament results for run C (see Table X) are unlike the previously seen tournaments, in so far as the strategy to perform best in the tournament is not that found at generation 500.

TABLE X
TOURNAMENT RESULTS FOR RUN C

	1	2	5	50	100	500
1	651	315	725	983	945	695
2	308	632	740	979	948	684
5	167	161	859	982	923	693
50	4	9	8	918	609	600
100	5	15	5	162	490	109
500	153	156	183	349	862	806

The best solution found at generation 100 considerably outperforms the best found at generation 500. Because we use coevolution when evolving strategies the fitness landscape is constantly changing. As our population changes so too may the attractors in our search space. This could lead to cycles during evolution to other areas of the search where robust solutions to our game exist.

While generation 500 is dominated entirely by generation 100, it is likely in response to the many changes in the fitness landscape that have occurred between the intervening generations. It is possible that generation 500 performs well against solutions found between generation 100 and 500 and that those strategies may outperform generation 100 in kind. To be sure this is what is happening, a round robin tournament between all strategies at all generations would need to be played which is extremely computationally expensive.

D. Unexpected Coordination Profiles and Control Profiles

Only one of our 20 runs of evolution returned unexpected data for both our coordination profiles and control profiles. We refer to this run as run D.

Once again, an early generation (generation 2 in this case) has a very high attack coordination, only to drop at generation 5 and again at generation 100. The best strategy found at generation 500 shows an increase once more, but not to as high a level as that seen at generation 2.

Our spread coordination profiles for run D (see Fig. 15) show an increase at generation 2 to as high as is seen throughout the run. This then drops at generation 5 and again at generation 100 similar to the attack coordination at those generations. This is somewhat similar to generation 2 in run B, where both attack and spread coordination appears to drop.

Our control profiles for run D (see Fig. 16) show that the rate at which territories are taken control of is at a maximum for this run at generations 1 and 2 over the first 10 turns of the average game but then drops off rapidly. Unlike run B, where the drop in both types of coordination was a result of learning a smarter spreading behavior, these results suggest that the strategy's ability to spread is good but that its ability to defend or attack (spreading into hostile territories) is poor.

The tournament results for run D (see Table XI) show how poor generations 1 and 2 actually are. Of the games played between the best strategies found at generations 1 and 2: generation 1 won 93, generation 2 won none and there were 907 draws. It can also be seen that when both strategies play themselves most games end in a draw. This shows that both strategies are not very aggressive (they did not attack often enough or they did not attack territories that were strategically important). At generations 5 and onward the number of games resulting in draws becomes very small as solutions become far more aggressive. Therefore the simultaneous drop in both attack and spread coordination is because of a poor attacking behaviour that was coordinated but not strategically clever.

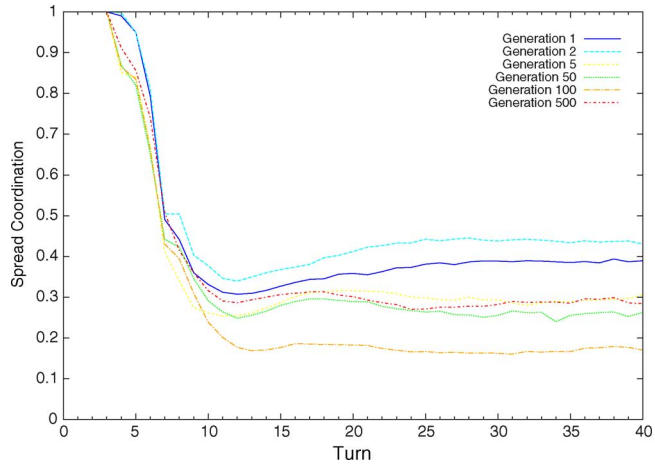

Fig. 15. Spread coordination for run D.

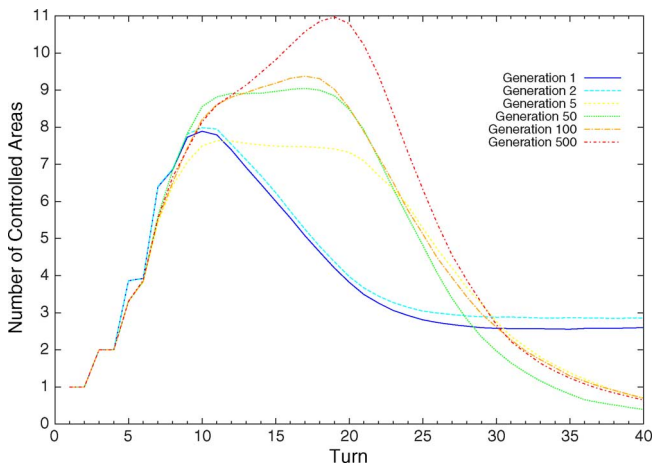

Fig. 16. Control profile for run D.

TABLE XI
TOURNAMENT RESULTS FOR RUN D

	1	2	5	50	100	500
1	5	0	988	982	988	984
2	93	177	993	989	996	994
5	11	6	1000	758	774	956
50	7	5	237	987	637	901
100	8	2	223	362	995	931
500	7	3	43	996	69	992

X. CONCLUSION

In summary, the approach shown in this paper demonstrates how to automatically generate strategies for a high-level RTS game (there is no need for a prior set of training strategies) and

that in general, as strategies evolve and become stronger, coordination is learned.

Twenty runs of coevolution were performed to learn strategies for an abstract real-time strategy game in a geographically static environment. A sample set of solutions was chosen from each run to best show the evolutionary path. Each sample set participated in a thousand round robin tournaments. We defined what we believe to be two essential types of coordination: spread coordination and attack coordination. Each of these measures of coordination were used to produce a coordination profile for each strategy as it plays an average game. We then analyzed how these profiles changed throughout the course of evolution to determine whether either type of coordination improved and the importance of each to a good strategy.

Firstly, it is important to note that neither of our measurements for attack and spread coordination imply a good attack or spread behavior. If an attack behavior lacks aggression, strategies will evolve towards a more aggressive behavior at the expense of either or both coordination measures. Also, if a strategy lacks a clever spreading behavior (it spreads to the same territory from different territories or does not stop often enough to take control of territories), solutions will evolve towards smarter behaviors at the expense of either or both coordination measures.

However, if the strategies within a population are aggressive and spreading in an intelligent manner then, in general, they will evolve good tactical attack coordination first at the expense of spread coordination, as aggression is very important to a winning strategy (as seen in run A). Then, when the attack coordination reaches a high level, does the spread coordination begin to recover. The spread coordination begins to increase with sometimes minor interferences with the attack coordination.

XI. Future Work

Our solutions are evolved in a static map environment and might therefore perform poorly when the map the game is played on is changed. Whether or not the efficacy of our evolved solutions are robust to changes in the spatial environment or specific to the map shown in Fig. 1 is unknown. In a future work we would like to analyze the effect of changes in the spatial environment on the solutions evolved in this paper in terms of coordination and overall effectiveness, and compare the results from these strategies against a set of strategies that has been evolved against a varying range of environments.

We wish to investigate how our automated player might learn to coordinate itself temporally, as at the moment our automated player plans only for the next immediate move in our game. We also hope to show how genetic programming can be used to learn cooperation between two or more automated players.

The eventual goal of our work is to apply our technique to an actual RTS game. Such games are more complicated than our current model represents. Specifically, the terrain may have certain strategic properties (e.g., increased cover, other resource types present, naval areas, impassable to land but not air, etc.) and there may be various types of units each with their own unique properties. But we feel that by adding to the nodesets in our GP and perhaps tweaking the plan creation process (e.g., by adding new order agent types such as "air," "land," and "naval" variants on the "move" and "attack" order agents already present), that creating a high-level plan for such a game is easily viable. It is also worth mentioning that while progressive refinement planning techniques, like that shown in this work, cannot be guaranteed to find an optimal plan, they are capable of returning good plans quickly. Therefore, we believe that this technique will scale up well, computationally, to more complex models of specific RTS games.

Acknowledgment

This work is being carried out with the support of the National University of Ireland, Galway. The authors would also like to thank the reviewers for their useful comments.

References

[1] K. Erol, J. Hendler, and D.S. Nau, "HTN planning: Complexity and expressivity," in *Proc. Nat. Conf. Artif. Intell.*, 1994, pp. 1123–1128.

[2] D. Andre and A. Teller, "Evolving team Darwin united," in *RoboCup-98: Robot Soccer World Cup II*, ser. Lecture Notes in Computer Science, M. Asada and H. Kitano, Eds. Berlin, Germany: Springer-Verlag, 1999, vol. 1604, pp. 346–352.

[3] S. Bakkes, P. Spronck, and E. O. Postma, "TEAM: The team-oriented evolutionary adaptability mechanism," in *Proc. 3rd Int. Conf. Entertain. Comput.*, 2004, pp. 273–282.

[4] A. L. V. Coelho, D. Weingaertner, R. R. Gudwin, and I. L. M. Ricarte, "Emergence of multiagent spatial coordination strategies through artificial coevolution," *Comput. Graph.*, vol. 25, pp. 1013–1023, 2001.

[5] N. Cole, S. J. Louis, and C. Miles, "Using a genetic algorithm to tune first-person shooter bots," in *Proc. Int. Congr. Evol. Comput.*, 2004, vol. 1, pp. 139–145.

[6] M. de Weerdt, A. ter Mors, and C. Witteveen, "Multi-agent planning: An introduction to planning and coordination," in *Proc. Handouts of the Eur. Agent Summer School*, 2005, pp. 1–32.

[7] D. Doherty and C. O'Riordan, "Evolving tactical behaviours for teams of agents in single player action games," in *Proc. 9th Int. Conf. Comput. Games: AI, Animat., Mobile, Edu. Serious Game.*, 2006, pp. 121–126.

[8] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computation*, 2nd ed. Berlin, Germany: Springer-Verlag, 2007.

[9] S. G. Ficici and J. B. Pollack, "Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states," in *Proc. 6th Int. Conf. Artif. Life*, 1998, pp. 238–247.

[10] K. D. Forbus, J. V. Mahoney, and K. Dill, "How qualitative spatial reasoning can improve strategy game AIs," *IEEE Intell. Syst.*, vol. 17, no. 4, pp. 25–30, 2002.

[11] G. Weiss, Ed., *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Cambridge, MA: MIT Press, 1999.

[12] R. Groß, K. Albrecht, W. Kantschik, and W. Banzhaf, "Evolving chess playing programs," in *Proc. Genet. Evol. Comput. Conf.*, 2002, pp. 740–747.

[13] S. J. Johansson, "On using multi-agent systems in playing board games," in *Proc. 5th Int. Joint Conf. Autonom. Agents Multiagent Syst.*, 2006, pp. 569–576.

[14] D. Keaveney and C. O'Riordan, "Analysing the fitness landscape of an abstract real-time strategy game," in *Proc. 9th Int. Conf. Intell. Games Simul.*, 2008, pp. 51–55.

[15] D. Keaveney and C. O'Riordan, Abstract Model of a Real Time Strategy Game Nat. Univ. Ireland, Galway, Ireland, Tech. Rep. nuig-it-011008, 2008.

[16] A. Lubberts and R. Miikkulainen, "Co-evolving a Go-playing neural network," in *Proc. Birds-on-a-Feather Workshop, Genet. Evol. Comput. Conf.*, 2001.

[17] C. Miles and S. J. Louis, "Co-evolving real-time strategy game playing influence map trees with genetic algorithms," in *Proc. Int. Congr. Evol. Comput.*, Portland, OR, 2006.

[18] J. Koza, "Genetic evolution and co-evolution of computer programs," in *Artif. Life II: Proc. Workshop Artif. Life*, C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, Eds. Reading, MA: Addison-Wesley, 1992, pp. 603–629.

David Keaveney received the B.Sc. degree in physics and astronomy from the National University of Ireland (NUIG), Galway, Ireland, in 2005.

He is currently a Researcher in the Department of Information Technology, NUIG. His current research focuses on cooperation and coordination in multiagent systems.

Colm O'Riordan received the B.Sc. and M.Sc. degrees in computer science from University College Cork, Ireland.

He lectures in the Department of Information Technology, National University of Ireland, Galway, Ireland. His main research interests are in the fields of agent-based systems, artificial life, and information retrieval. His current research focuses on cooperation and coordination in artificial life societies and multiagent systems.