

Evolving Robust Strategies for an Abstract Real-Time Strategy Game

David Keaveney and Colm O’Riordan

Abstract—This paper presents an analysis of evolved strategies for an abstract real-time strategy (RTS) game. The abstract RTS game used is a turn-based strategy game with properties such as parallel turns and imperfect spatial information. The automated player used to learn strategies uses a progressive refinement planning technique to plan its next immediate turn during the game. We describe two types of spatial tactical coordination which we posit are important in the game and define measures for both. A set of ten strategies evolved in a single environment are compared to a second set of ten strategies evolved across a set of environments. The robustness of all of evolved strategies are assessed when playing each other in each environment. Also, the levels of coordination present in both sets of strategies are measured and compared. We wish to show that evolving across multiple spatial environments is necessary to evolve robustness into our strategies.

Index Terms—Genetic Programming, Real-Time Strategy, Generalisability, Coordination

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 only 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 for military and economic purposes.

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 pieces than these classical games and require that all pieces of all players are moved in parallel. The branching factor of such games is far larger than most classical board games and makes forward based reasoning approaches (such as traditional minimax algorithms) impractical for planning.

While automated players in RTS games today often exhibit a good tactical behaviour, 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) [12]. First, the problem is solved at the strategic level using high-level actions. Each high-level action is then solved using lower level actions.

David Keaveney, Postgraduate Student, National University of Ireland, Galway. (E-mail: d.keaveney1@nuigalway.ie)

Colm O’Riordan, Lecturer in the Dept. of IT, National University of Ireland, Galway. (E-mail: colm.oriordan@nuigalway.ie)

This process continues for each unsolved abstract action until it is solved using in-game (atomic) actions.

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[8]. Such abstraction methods are useful for high-level planning. These spatial representations are similar to how space is represented in many turn-based strategy board games. Because this work only concerns learning high-level strategies the test-bed game used in this paper is an abstract model of a real-time strategy game. High level strategies could be learned using this abstract model and then implemented in a RTS game such as ORTS [2] using a HTN to decompose the high level plan into in-game actions.

In this paper, two distinct sets of strategies for an abstract RTS game are co-evolved. An abstract representation of a RTS game is used primarily because we only wish to learn coordination at this level of planning. It also reduces the computational expense associated with using evolutionary computation techniques and renders the results more amenable to analysis. The first set of strategies is evolved for one specific environment while the other set is evolved across multiple environments. These evolved strategies utilise a multiagent system framework and a progressive refinement planning technique that allow them to plan their next immediate set of actions. We describe two types of coordination we feel are important in the game and define measurements for both. Both sets of evolved solutions are compared to identify if there are any differences in their levels of coordination throughout an average game. Also, the effectiveness of all strategies against each other is measured across our set of multiple environments.

The aim of this work is to show how evolutionary programming techniques, specifically genetic programming, can be used to automatically generate strategies for real-time strategy games. We also wish to show that evolving these solutions across multiple map environments will lead to more robust strategies.

Section II introduces the field of computational techniques known as evolutionary computation. Section III briefly discusses other research in the area of evolutionary computation with respect to spatial decision making and multiagent system approaches to strategic board-games. 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 behaviour 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. We outline in section VII the two types of coordination we believe are important in the test-bed game and then define general measurements for both. Section VIII describes the experiments performed to evolve solutions more robust to changes in the environment by evolving strategies in multiple spatial environments and then describes the means by which we compare these strategies to others evolved in a specific environment. 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. INTRODUCTION TO EVOLUTIONARY COMPUTATION

Evolutionary computation (EC) describes computing techniques inspired by biological evolution. EC techniques are computationally expensive and are not often used in commercial game development. However, these techniques have been used effectively in games such as Risk [9] and Diplomacy [9] (high-level strategy games similar to the test-bed used in this work) to real-time games such as RoboCup [3], [1] and first person shooter (FPS) games [4], [5]. 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 [6]. 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 can be created by combining the fittest of individuals found in the current generation. These new solutions can be constructed using recombination 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 optimise computer programs based on their fitness. An important property of GP is that it can search through large search-spaces efficiently. Furthermore, the GP representational language uses a computer program structure which means that neither the size of a solution nor its shape needs to be constrained beforehand, as is the case with other similar techniques such as genetic algorithms. This property of GP allows for a much broader search thereby increasing the potential for better solutions.

Usually solutions within a population receive a fitness by being evaluated against an already known static solution. However, when using co-evolution, solutions within the same population play against each other. This means that their fitness depends on how well they compete with other solutions within the population. Because of this relative measure of fitness, once fit solutions, may become unfit in subsequent generations. "Co-evolution has been proposed as a way to evolve a learner and learning environment simultaneously

such that progress arises naturally with minimal inductive bias." [7]

III. RELATED WORK

Stefan Johansson and Fredrik Haard [9] 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 Chris Miles and Sushil J. Louis [13] 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 co-evolved solutions were used directly as operational controllers rather than for strategic planning and produced behaviours that were as effective as their hand-coded strategies only more robust.

IV. ABSTRACT REAL-TIME STRATEGY GAME

The abstract RTS game used in this paper is called 'Bellum Bellum Gratia' (BBG) and was designed for high-level RTS game research [10]. It is a turn-based strategy game in which turns are resolved in parallel which models a real-time game. This type of turn resolution can be seen in the popular strategy board game 'Diplomacy'. Unlike any board game, BBG also has imperfect spatial information of the environment similar to the effect known as 'fog of war' in RTS games. 'Fog of war' describes the area of a map that can not currently be seen by any units and hides any dynamic information such as enemy unit or building positions within its area of effect.

The game uses an undirected graph to describe the structure of the environment. Vertices represent territories of which there is only one type and edges define borders between territories. All players receive a home territory which they control and receive a set number of units at that territory at the start of the game. All units are equal and are produced at a player's home territory. All units can move across one edge per turn. Units occupying uncontrolled areas for a turn receive control of that territory and controlling territories gives a player increased unit support. Any unit occupying an enemy controlled territory for a turn renders that territory uncontrolled by any player. On each turn, a player receives up to two units at its home territory until it reaches the maximum number of units it can support.

Should two adjacent occupied areas, one of which is hostile to the other, move units to the other territory, an attack occurs. No units actually move into either territory but the player considered to have the least strength in the

conflict loses a certain number of units that participated in the battle. Each additional territory involved in an attack against another territory provides a flanking bonus to the strength of that attack. There are more complex factors that are used to determine the strength of a player in any move or attack which stem from the parallel order resolution property of the game. A full description of these factors are omitted from this paper but can be found in the technical report describing this test-bed game [11].

V. MAS AUTOMATED PLAYER

When creating a plan for the next immediate turn in our game our automated player creates order agents for all occupied areas and all areas surrounding those areas (either an attack order agent or move order agent). Each of these potential orders are then assessed using a function we refer to as the ‘Order Minimum’ (OM) function which determines the number of units required for that order to be considered a success.

Next begins a progressive refinement planning technique, whereby the current plan is created incrementally by committing one unit at a time to an order from the possible set of valid orders. All the orders are evaluated using a function termed the ‘Order Priority’ (OP) function. This function’s purpose is to locate an order that will be allowed to commit one unit to itself. When this order has been found, another function, the ‘Edge Priority’ (EP) function, assesses all the edges belonging to that order. These edges point towards all surrounding areas with units that can be committed and include an identity edge that points towards the order’s own area. The edge with the highest priority has the area 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.

At this stage, all orders are once again evaluated using the ‘Order Priority’ 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 ‘Order Minimum’ function). Any failed order found (an order that does not have the required units necessary for success) is removed for the set of possible orders and has any units they had committed to themselves released of there commitments. Should this happen, this final order checking stage ends and the previous refinement planning stage begins again with the newly released units left to commit and one fewer order agent to consider.

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

VI. GENETIC PROGRAMMING

Genetic programming (GP) is used to evolve strategies for our abstract RTS game. Three functions determine the behaviour of the MAS automated player: the ‘order minimum’ function, the ‘order priority’ function and the ‘edge priority’ function. The genotype for all individuals contains three different trees, each corresponding to one of the three functions. This allows all three functions to evolve in parallel.

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

TABLE I
FUNCTION NODESET

Tables I and II show all the function nodes and terminal nodes we use when evolving 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’¹. Each of these nodes can belong to one or more node-sets. To constrain the creation of trees to valid programs each function node has rules for each of their children determining what node-sets are valid in choosing a child node.

Description of Node	OM	OP	EP
Enemy Strength	x	x	
Friendly Support		x	x
Enemy Support	x	x	
Committed		x	x
Non Committed (or Fr. Strength)		x	x
Friendly Connectivity	x	x	x
Enemy Connectivity	x	x	x
Unknown Connectivity	x	x	x
Minimum Needed		x	x
Order Priority			x
Surrounding Friendly	x	x	x
Surrounding Enemy	x	x	x
Distance to Fr. Homebase		x	x
Distance to En. Homebase		x	x
Constants 1,2,3...50,75,100	x	x	x
Conditional Nodes	OM	OP	EP
Enemy Homebase	x	x	x
Friendly Homebase	x	x	x
Enemy Controlled	x	x	x
Friendly Controlled	x	x	x
Neutral Controlled	x	x	x
Attack Order	x	x	

TABLE II
TERMINAL NODESET

A. Co-evolution and Fitness Evaluation

Each member of the population plays a set number of games against other members of the population chosen at

¹OM = Order Minimum Function, OP = Order Priority Function, EP = Edge Priority Function

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.

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

$$Fitness_{win} = T_{end}$$

$$Fitness_{draw} = (1.2 * T_{max}) - (Controlled * 0.2)$$

$$Fitness_{loss} = (2.2 * T_{max}) - T_{end}$$

The overall fitness of a solution is also penalized for length. For each node in a solution a length penalization factor is added to the fitness (low values for fitness represent fit solutions). 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 fitness 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. A sub-set of individuals are chosen from the evaluated population at random. The size of this sub-set is called the tournament size. The fittest solution within this sub-set is the selected individual. Therefore, 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 described below:

1) *Crossover Operator*: The crossover operator used during the evolutionary process first selects one of the three sub-trees present in all solutions. Parent ‘A’ then has a random node on that sub-tree selected. A node is then chosen randomly on Parent ‘B’. If this node is not of the same node-set as that of the node chosen from parent ‘A’ another node is randomly chosen. When both selected nodes are of the same node-set, they are swapped with each other, including all descendant nodes. The two resultant trees are now the children of parent ‘A’ and parent ‘B’.

2) *Single Node Mutation Operator*: This type of mutation operator selects a random node from any of the three sub-trees. It 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 selects a random node from any of the three sub-trees. This node is then replaced with a new sub-tree, 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

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

In order to spread quickly, it is important to move into as many uncontrolled areas as is possible. This type of coordination is termed ‘spread coordination’ and it is this type of coordination that should dominate the opening game-play of successful strategies.

Later in the game when both players meet, attacks can occur. As attacks benefit from the number of separate territories involved in the attack (i.e. each additional territory involved in an attack after the first provides a bonus to the attack’s strength), 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. Tactical Spread Coordination Measure:

This global measure of spread coordination assesses how well a strategy has moved units into as many uncontrolled areas as is immediately possible during a turn in our game.

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

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

B. Tactical Attack Coordination Measure:

This global measure of attack coordination determines how well a strategy utilises as many friendly occupied areas surrounding attacked areas in those attacks and therefore,

how well it avails of the flanking bonus provided by extra areas involved in an attack.

A	Set of attacked areas
B_i	Set of surrounding friendly occupied areas of area i
C_i	Set of areas involved in attack on area i

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

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

VIII. EXPERIMENTS

In preliminary work, we saw how GP can be used to evolve a fit strategy by co-evolving populations of strategies playing on the one map. However, we did not know how robust these strategies were and if the strategies had learned abilities that were useful on other maps. The purpose of these experiments is to analyse the differences between solutions evolved for our abstract real-time strategy game using one spatial environment (i.e. the graph structure the game is played on) in the fitness function and those evolved using multiple such environments/maps.

Ten runs of evolution are performed for our abstract RTS game using one specific map for all games played and another ten runs of evolution where a set of five maps are used to evaluate fitness during evolution. The GP parameters for all of these runs can be seen in table III.

TABLE III
GP SETTINGS

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

The best strategy at generation five hundred is saved for each run. This provides us with ten solutions evolved for one particular map (shown in Figure 1) and another ten solutions evolved for five particular maps (shown in Figures 1 and 2). The former set of solutions we refer to as set A and the latter set of ten solutions we call set B. The strategies in ‘set A’ are numbered ‘1’ through ‘10’ and those found in ‘set B’ are numbered ‘11’ through ‘20’.

Each hexagon in Fig. 1 and Fig. 2 represents an area of the map and those with sides adjacent to each other

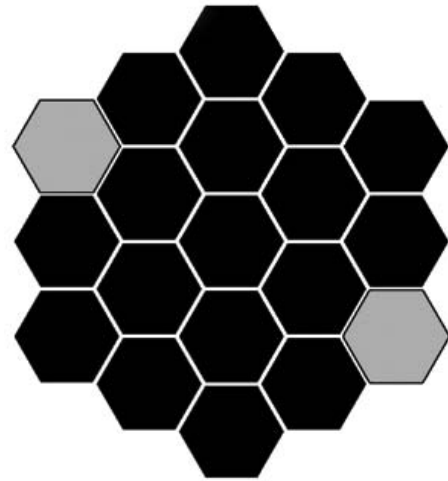


Fig. 1. Map 1

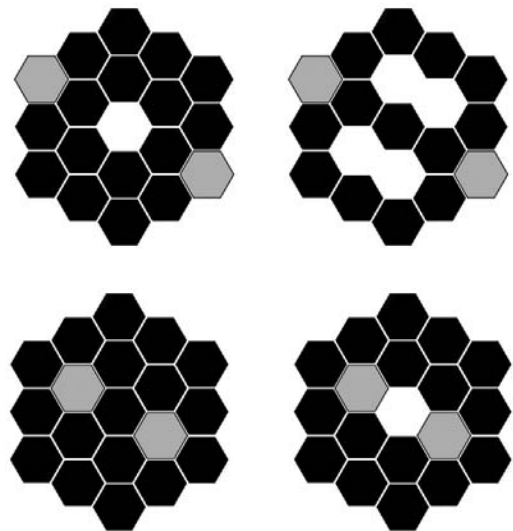


Fig. 2. Map 2(top left), Map 3(top right), Map 4(bottom left), Map 5(bottom right)

share a connecting edge. The two shaded areas represent the homebases (starting locations) for both opposing players.

One thousand round-robin tournaments are performed with all twenty strategies that comprise set A and set B for each of the five maps used when evolving set B (i.e. each strategy will play every other strategy one thousand times on each map). Each win will give a score of 1.0 and each draw gives a score of 0.5. This will determine the positions of each solution in each league. All five leagues are then summed together to determine the efficacy and robustness of each strategy when played over all maps.

During each league an attack coordination profile and spread coordination profile is created for each player-player pairing (how coordinated a player is against another player). During each game, at every turn, for each player a measure of the coordinated spread and coordinated attack values (if applicable) are recorded. Average values of these measures

are maintained.

These player-player profiles are then averaged together for each player/strategy to get a generalised coordination profile for each strategy (i.e. how coordinated a strategy is against all the other nineteen strategies found in sets A and B). In turn, the generalised coordination profiles for each strategy are averaged together across all leagues to produce profiles that show how coordinated a strategy is against all others strategies when played across all five maps. These averaged profiles are then compared to each other in order to evaluate how these measures of coordination change throughout the average game.

A. Coordination Measurement Analysis

As attack coordination is not applicable in the first few opening turns of the game we expect to see high values of spread coordination. After the second turn in our game, attacks become 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 the initially high spread coordination begin to drop as attacks are more likely to hamper the spreading behaviour.

From previous experience with evolving strategies for our game [10], 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 (depends on the map environment and opponent strategy) to the end of the game.

IX. RESULTS

A. Performance

To determine the effectiveness and robustness of all the evolved strategies across all five map environments we analyse the tournament results obtained from summing together five tournaments, each of which were played on one of the five maps. We then rank each strategy from highest score to lowest. The rank of each strategy when played across all maps can be seen in Fig. 3, together with the rank of each strategy in tournaments played on individual maps.

The top five positions in our tournament played across all maps are held by solutions from set B (i.e. those evolved on all five maps). This shows that solutions evolved on 'Map 1' (i.e. set A) lack sufficient generalisability in their behaviour compared to those evolved across all five maps (i.e. set B).

When showing the results for all tournaments (see Tables IV- IX), we concentrate on six solutions in particular: the three best performing and robust strategies from 'set A' (i.e. '2', '4' and '8') and the three best scoring and robust strategies from 'set B' (i.e. '17', '15' and '16').

As is clear from the results of the overall tournament played over all five maps (see Tab. IV), strategies from 'set A' do not entirely dominate those from 'set B'. Solution '17' is the highest scoring strategy overall yet the best from 'set A' (Solution '2') still takes 6th position.

In order to get a better understanding of these results, we analyse the individual tournaments each played on one of the five maps (see Tables V- IX for partial results).

	AllMaps	Map1	Map2	Map3	Map4	Map5
1st	17	6	16	17	17	15
2nd	15	7	15	14	20	17
3rd	16	8	14	11	19	16
4th	18	4	6	18	13	20
5th	14	2	2	15	2	4
6th	2	10	8	16	18	13
7th	4	16	18	4	7	18
8th	20	1	4	2	16	7
9th	8	15	5	8	14	14
10th	6	3	1	5	15	19
11th	5	18	17	12	12	5
12th	7	5	10	13	5	8
13th	19	14	3	19	6	6
14th	13	12	12	6	11	12
15th	12	17	20	7	8	1
16th	10	13	11	20	1	11
17th	11	20	19	3	10	10
18th	3	9	13	10	4	2
19th	1	19	9	9	9	9
20th	9	11	7	1	3	3

Fig. 3. Strategy Ranks for all Tournaments

TABLE IV
PARTIAL RESULTS FOR OVERALL TOURNAMENT (ALL 5 MAPS)

	..2..	..4..	..8..	..15	16	17..
..2..	-	2426	1935	3731	2191	3483
..4..	2574	-	2323	2959	2179	3521
..8..	3065	2677	-	2735	3234	3339
..15	1269	2040	2264	-	2880	2033
16	2809	2820	1765	2119	-	3072
17..	1517	1479	1660	2966	1927	-
All	51266	50880	50118	59496	57133	62093
Rank	6th	7th	9th	2nd	3rd	1st

As can be seen from Fig. 3, the six best performing strategies on 'Map 1' all belong to set A. This is to be expected since all strategies from 'set A' were evolved specifically for this map environment. The three overall best strategies from 'set A' rank third, fourth and fifth (see Tab. V). Therefore, while solutions '6' and '7' are better than solutions '2', '4' and '8' on the map they were evolved, these latter solutions learned a more general behaviour allowing them to perform better across all five maps.

The tournaments played on Maps 2 - 5 show strategies from 'set B' generally outperforming those from 'set A' (with a few exceptions) with no specific strategy from 'set A' ever outperforming the best from 'set B'. The tournament played using 'Map 4' clearly shows 'set A' have difficulty competing

TABLE V
PARTIAL RESULTS FOR TOURNAMENT ON MAP 1

	..2..	..4..	..8..	..15	16	17..
..2..	-	581	448	424	388	179
..4..	419	-	541	473	403	310
..8..	552	458	-	398	401	340
..15	576	527	602	-	466	332
16	612	596	599	533	-	209
17..	821	690	660	668	791	-
All	11943	11952	12040	10352	10878	7447
Rank	5th	4th	3rd	9th	7th	15th

TABLE VI
PARTIAL RESULTS FOR TOURNAMENT ON MAP 2

	..2..	..4..	..8..	..15	16	17..
..2..	-	510	321	610	652	621
..4..	491	-	509	489	571	577
..8..	679	491	-	788	647	350
..15	391	512	212	-	443	244
16	349	429	353	557	-	398
17..	380	423	651	756	602	-
All	11425	10682	11393	12248	13078	9966
Rank	5th	8th	6th	2nd	1st	11th

with ‘set B’ with strategy ‘17’ winning all games against shown opponents and nearly all strategies from ‘set A’ being ranked in the bottom half. It can also be observed that while strategies ‘2’, ‘4’ and ‘8’ score well in total over all maps, they perform very poorly on certain maps. Strategies ‘4’ and ‘8’ rank eighteenth and fifteenth respectively on Map 4 and Strategy ‘2’ rank’s eighteenth on Map 5.

TABLE VII
PARTIAL RESULTS FOR TOURNAMENT ON MAP 3

	..2..	..4..	..8..	..15	16	17..
..2..	-	590	553	921	246	916
..4..	410	-	439	569	494	831
..8..	447	562	-	600	759	904
..15	80	431	400	-	589	300
16	755	506	241	412	-	672
17..	85	170	96	700	329	-
All	10348	10696	10303	12332	11871	14879
Rank	8th	7th	9th	5th	6th	1st

TABLE VIII
PARTIAL RESULTS FOR TOURNAMENT ON MAP 4

	..2..	..4..	..8..	..15	16	17..
..2..	-	0	2	849	0	1000
..4..	1000	-	313	886	385	1000
..8..	999	688	-	161	671	1000
..15	152	115	839	-	921	1000
16	1000	616	329	80	-	1000
17..	0	0	0	0	0	-
All	10969	6275	6677	9359	9487	16500
Rank	5th	18th	15th	10th	8th	1st

TABLE IX
PARTIAL RESULTS FOR TOURNAMENT ON MAP 5

	..2..	..4..	..8..	..15	16	17..
..2..	-	746	612	929	906	768
..4..	255	-	522	544	327	804
..8..	389	479	-	789	757	746
..15	72	457	212	-	462	158
16	94	674	244	538	-	794
17..	232	197	254	843	206	-
All	6581	11277	9705	15207	11819	13302
Rank	18th	5th	12th	1st	3rd	2nd

B. Tactical Coordination

Fig. 4 shows the attack coordination profiles for all twenty evolved strategies. Those evolved for Map 1 specifically have a high tactical attack coordination score. However, only some

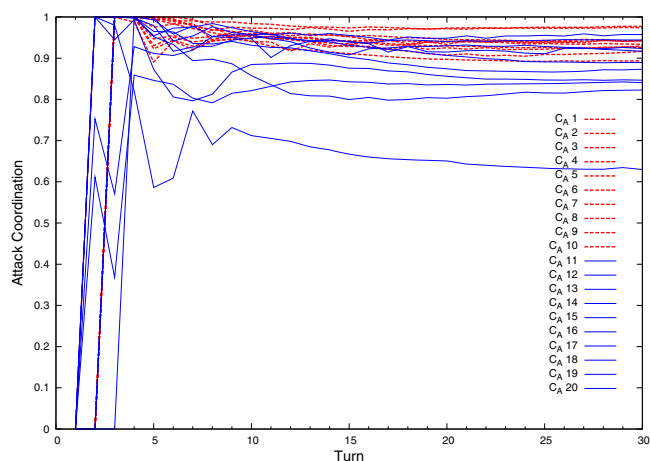


Fig. 4. Attack coordination profiles for all strategies against all other strategies over all environments

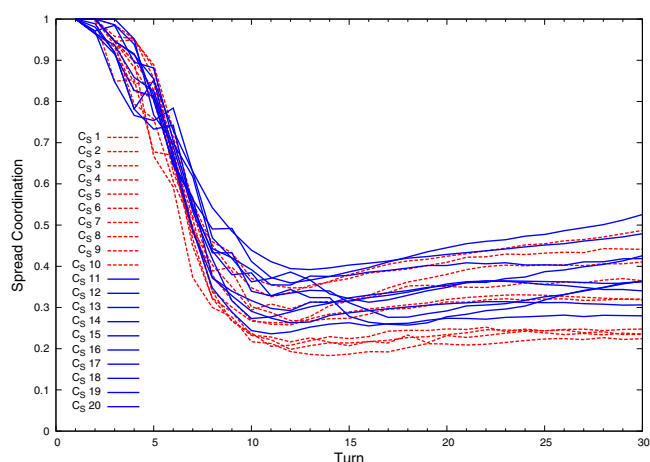


Fig. 5. Spread coordination profiles for all strategies against all other strategies over all environments

of the strategies evolved across all five map environments have a high degree of tactical attack coordination similar to ‘set A’, with the other strategies being coordinated less so. This shows that evolutionary runs with a changing map environment in the fitness function find it harder to evolve good tactical attack coordination.

Fig. 5 shows the spread coordination profiles for all twenty of the evolved solutions. A few solutions from ‘set A’ have profiles with poorer tactical spread coordination than any seen from ‘set B’. This is a consequence of having a very high attack coordination that causes interference in their ability to spread.

Fig. 6 shows attack coordination profiles (above the legend) and spread coordination profiles (below the legend) for the top three most robust strategies from ‘set A’ and from ‘set B’. All of the robust strategies from ‘set B’ have very good tactical attack coordination profiles, comparable to the high attack coordination seen for strategies in ‘set A’. Spread coordination profiles for all six strategies are similar.

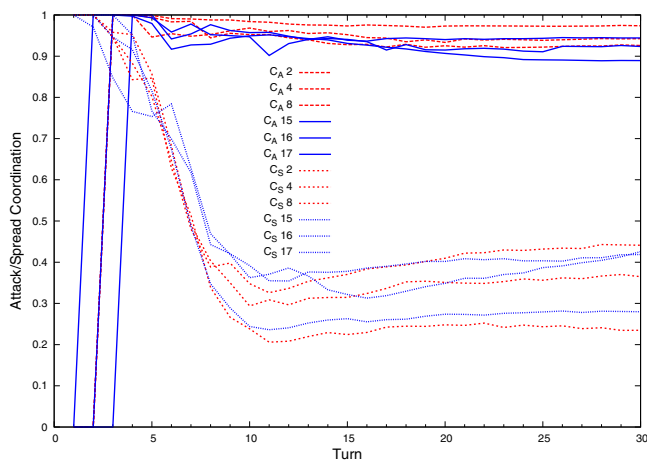


Fig. 6. Attack and spread coordination profiles for our six most robust strategies against all other strategies over all environments

X. CONCLUSIONS

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 highlights the differences in the robustness of solutions evolved for a single environment and those evolved across multiple environments.

Two sets of ten runs of co-evolution were performed. The first ten runs are evolved for one spatial environment and the second ten runs are evolved over a set of five different environments. Finally, all twenty strategies took place in one thousand round-robin tournaments on each of the five map environments the results of which detail the effectiveness of each strategy against each other strategy in each environment. Attack and spread coordination information was also recorded for each strategy pairing in each environment.

From analysing the results of these tournaments, it is clear that the first set of solutions (i.e. 'set A') are not so specialised that they are always beaten by those of the second set of solutions (i.e. 'set B') in all environments other than Map 1. Yet, in all tournaments in environments other than Map 1, the highest scoring strategies are those from 'set B'. Therefore solutions evolved on Map 1 alone possess enough generalised behaviour to perform adequately well in other environments but are not as robust to changes in the environment, in general, as solutions evolved across multiple different maps.

The coordination profiles for all strategies suggest it is harder to evolve a high attack coordination across multiple environments than on one environment and that a high attack coordination rather than spread coordination is important to an effective robust strategy.

XI. FUTURE WORK

As follow up work, to further assess the robustness of our solutions, we will need to test the evolved strategies on different map environments other than those used during evolution. Also, the true quality of our evolved solutions

is not known because they are only evaluated relative to each other. As there exists no known strong strategy to act as a baseline and other traditional minimax methods are impractical, as future work, we will be subjecting our evolved solutions to human evaluation.

In future work, we wish to investigate how our automated player might learn temporal coordination, as currently our automated player plans only for the next immediate turn in our game. We also hope to show how genetic programming can be used to learn cooperation within teams of two or more automated players.

Our planning method might also be used as a strategy identification method. As such, would we like to be able to evolve a diverse set of robust solutions that could then be used in both identifying an enemy strategy and in selecting an appropriate strategy with which to respond for optimal effect.

Eventually, we plan to implement this approach with an actual real-time strategy game (e.g. ORTS), by tuning the abstract model to represent the RTS game in question and developing a hierarchal task network (HTN) to decompose the produced high-level plans into low-level plans.

REFERENCES

- [1] D. Andre and A. Teller. Evolving team darwin united. In *RoboCup-98: Robot Soccer World Cup II, LNCS 1604*, M. Asada and H. Kitano, eds., Springer-Verlag, pages 346–352, 1999.
- [2] Michael Buro. Orts: A hack-free rts game environment. In *Computers and Games*, volume 2883/2003 of *Lecture Notes in Computer Science*, pages 280–291. Springer Berlin / Heidelberg, 2003.
- [3] Andre L.V. Coelho, Daniel Weingaertner, Ricardo R. Gudwin, and Ivan L.M. Ricarte. Emergence of multiagent spatial coordination strategies through artificial coevolution. In *Computer & Graphics*. N. 25, pages 1013–1023, 2001.
- [4] N. Cole, S. J. Louis, and C. Miles. Using a genetic algorithm to tune first-person shooter bots. In *Proceedings of the International Congress on Evolutionary Computation*, 2004.
- [5] D. Doherty and C. O'Riordan. Evolving tactical behaviours for teams of agents in single player action games. In *Proceedings of the 9th International Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games*, pages 121–126, 2006.
- [6] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computation, 2nd Edition*. Springer, 2007.
- [7] Sevan G. Ficici and Jordan B. Pollack. Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In *Proceedings of the Sixth International Conference on Artificial Life*, 1998.
- [8] K.D. Forbus, J.V. Mahoney, and K. Dill. How qualitative spatial reasoning can improve strategy game ais. In *IEEE Intelligent Systems*, pages 17(4):25–30, 2002.
- [9] Stefan J. Johansson. On using multi-agent systems in playing board games. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems AAMAS '06*, 2006.
- [10] D. Keaveney and C. O'Riordan. Analysing the fitness landscape of an abstract real-time strategy game. In *Proceedings of the Ninth International Conference on Intelligent Games and Simulation (GameOn)*, pages 51–55, 2008.
- [11] David Keaveney and Colm O'Riordan. Abstract model of a real time strategy game. Technical report, National University of Ireland, Galway, 2008.
- [12] Jasper Laagland. A htn planner for a real-time strategy game. <http://hmi.ewi.utwente.nl/verslagen/capita-selecta/CS-Laagland-Jasper.pdf>.
- [13] C. Miles and S. J. Louis. Co-evolving real-time strategy game playing influence map trees with genetic algorithms. In *Proceedings of the International Congress on Evolutionary Computation, Portland, Oregon*. IEEE Press, 2006.