Coevolving team tactics for a real-time strategy game

Phillipa Avery, Sushil Louis

Abstract— **In this paper we successfully demonstrate the use of coevolving Influence Maps (IM)s to generate coordinating team tactics for a Real Time Strategy (RTS) game. Each entity in the team is assigned their own IM generated with evolved parameters. The individual IMs allows each entity to act independently of the team and team coordination is then achieved by evolving all team entities' IM parameters together as a single chromosome with a single evaluation. These evolved parameters are then evaluated by measuring performance against another coevolving population of individuals. Using this method we have generated some interesting strategies, and have demonstrated the potential of using IMs for coevolving team tactics. In the future, coevolved tactics could then be used to provide a challenging opponent for human players.**

I. Introduction

This paper investigates the coevolution of boat group tactics in a real-time capture-the-flag game. We present a method of evolving group tactics using spatial information to specify the relative positions of entities, and for maneuvering of entities to achieve a set of objectives. In this research we are interested in generating new and interesting tactics and eventually applying these tactics to playing against human players.

The ability to automatically generate spatially resolved group tactics has applications in computer gaming, robotics, multi-agent systems, and war-gaming. Our prior work in coevolving real-time strategy game players pointed out the importance of tactical superiority when strategies and resources are balanced among competing players [1]. Real-time strategy games tend to be non-linear with early decisions having a large influence on final outcome. Bad tactical planning can ruin good strategy and good tactics can mitigate the ill effects of mediocre strategy. Tactical planning thus plays an important role in many situations.

We define strategic planning as the process of generating a plan that uses all available game resources to win. This generated plan typically has a set of intermediate objectives that must be achieved in order to win. Tactical planning then is the process of generating plans that can be used to achieve these individual strategic objectives using a subset of available resources.

The need for a good distributed controller for RTS games is obvious, but non-trivial. The non-linear nature of the independent entities that are working to act together to achieve a common objective makes writing such a controller difficult. One possible way of achieving such a controller is through the use of evolutionary computing techniques such as Genetic Algorithms (GA)s. These techniques were specifically designed for such non-linear problems, and hence appear to be well suited to the generation of a distributed controller. Therefore, we wish to apply a GA to design automated, adaptive, tactical controllers. For our research, we implemented a coevolutionary algorithm to create such a tactical controller for small groups of boats. We look to create a controller that will work towards tactics involving deception, sacrifice, feints, flanking, splitting enemy firepower, and possibly others that we have not foreseen. In our case, we want a controller mechanism that will allow the game entities (boats) to function autonomously, while still coordinating with the rest of the team to achieve the team objectives. By working autonomously, the boat has independence to move and attack on its own, while the coordination aspect allows the boat to use its autonomous behaviour towards the team's objectives.

To achieve the controller mentioned, we present a representation for generating adaptive tactics using coevolved Influence Maps (IMs). The autonomous behaviour is achieved through the use of an IM for each entity. As the IM changes with the state of the game, this representation allows for adaptive tactic generation. The use of an IM for each entity allows an entity to move autonomously, generating their paths independently from the rest of the team. To then achieve the coordination of tactics, all the IM parameters for a team are evolved together. This allows an evolutionary individual to represent autonomous IMs for a team, that work together to achieve the objectives defined in the evaluation function. In this way, the coevolutionary algorithm can find IMs that work together to coordinate tactics.

In our representation, an IM is generated using evolved parameters. The parameters represent influence weights for each entity in the game, which are then used by the IM function to create the IM. An entity then uses the IM to find a goal and path. The goal is defined as the most attracting weight on the IM, and the path is then created by calculating the A* path through the influence map to the goal.

In this paper, we explain the mechanisms of this representation and the coevolution, and demonstrate the success of the technique in our experiments. We highlight the success of the technique to create some complex tactical manoeuvres. The paper is organized as follows. Section II provides background, including relevant subject matter and other research. We also describe the previous work done by the authors on this research. Section III describes the changes made to implement a coevolutionary approach, and the methodology of how this was implemented. Section IV gives the experiments and results for this research. Finally, section V concludes the paper with a discussion on the findings and future work.

Phillipa Avery and Sushil Louis are with the Department of Computer Science and Engineering, University of Nevada, Reno, USA (email: pippa@cse.unr.edu, sushil@cse.unr.edu).

II. Background

This section introduces IMs and our reasoning for using coevolution. Following this we identify work related to the research. Finally, we provide a brief summary of the research done previously on this topic by the authors.

A. The use of influence maps and coevolution

An Influence Map (IM) is a useful technique to represent spatial interest points in a game. The IM is applied as a graph or grid placed over the map. Any points of interest on the map can be highlighted, and the map broken into discrete regions. The IM function then assigns values (weights) to each node or cell in the IM. These weights can identify boundaries of control, "choke points", and other tactically interesting spatial features.

Originally IMs evolved out of work done on spatial reasoning within the game of Go [2], and have since then been occasionally used in video games and the AI and CI in games communities [3], for games such as Pacman [4], real-time strategy games [1], [5]–[8], and turn-based games [9]. Typically the use of an IM in RTS style games, involves generating the IM at each time step in the game (using the IM function), and then using the assigned weights for path finding. As the game state changes, e.g. resources are used up or game entities move or are destroyed, the IM function that assigns the weights will create a new IM reflecting these changes.

There exist many possible representations that could be used for spatial tactics, such as artificial neural networks and knowledge based approaches. The reason we chose to use an IM representation is as follows. IMs give an intuitive representation of spatial characteristics, and as a result have been used extensively for spatial reasoning in games as discussed above. IMs also allow us to easily scale for complexity, with the addition of new game entities. Finally, IMs allow us easy visualization and interpretation of the evolved tactics.

As mentioned previously, we chose a coevolutionary algorithm to create our controller. This allows generation of new tactics through self-discovery. The use of coevolution has a successful history for self-learning strategies in game playing [10]–[12]. By using this ability of coevolution, we hope to generate new and interesting tactics for the naval RTS games. Eventually we also wish to use the adaptive nature of coevolution to generate tactics that can adapt to a human player.

B. Related work

Previous work has been done in our lab to evolve a single influence map **tree** to generate objectives for the LagoonCraft RTS game [13]. The objectives were achieved through a genetic algorithm that generated near-optimal allocation of resources [1]. The research described in this paper differs in that we are evolving independent co-operative influence maps for a group of entities that coordinate to complete a task. Each entity uses an individually assigned IM, but the fitness evaluation assigns one fitness to the entire set of influence maps evolved. In other words, if we have five boats in our attack group, each individual in the population contains parameters that generate five influence maps.

Other work using IMs for game playing has been done by other research. We summarize this work here. Sweetser and Wiles [14] used IMs for game agents, where the IM was used to create reactive agent behaviour to environmental changes. While we are looking at group tactics over agent behaviour, this research demonstrates the benefits of IMs in agent decision-making. Bergsma and Spronck also use influence maps to generate tactics for a turn based strategy game [9]. A neural network layers several IMs to generate attack and move values for each cell. Like Miles' work, Bergsma and Spronck combine IMs to generate tactical or strategic objectives. Unlike Miles' work, they use neural networks (whose weights are evolved) to combine the influence maps. Similar work has been done by Jang and Cho who also use layered IMs with a neural network for a real-time strategy game [15]. While we are researching games, specifically real-time strategy games, it is worth noting that the use of IMs for spatial decision making also has benefits in areas such as robotics [16]. Danielsiek et. al. [17] use IMs in conjunction with flocking to determine movement of groups in RTS games, with the goal of cohesively moving groups of units. Our goal differs in that we are attempting to generate autonomous tactics for each entity, which are evolved to coordinate their tactics.

While the work summarized above is most relevant to our research, there is a large variety of other work using influence maps for spatial reasoning, organizational behaviour, and in other non game related fields. To the best of our knowledge, our research is the first to co-evolve a set of co-operative influence maps together, and use these co-adapted plans to generate competent, adaptive tactics for attack and defense.

C. Previous work

The research in this paper is a continuation of work published in [18]. The previous work demonstrated the use of evolving and coevolving Influence Maps (IMs) for a team of entities in a real-time strategy scenario. This research extends the previous work by further investigating the use of coevolution with the IM representation. This section provides a summary of the previous research, as a precursor to this research.

In the previous research, we used a capture-the-flag game with two teams of boats (entities), each with their own goal. The first team, the defenders, had the goal of protecting a flag for the given time limit. The other team, the attackers, had to reach the opponent's flag within the time limit, with sub-goals of destroying as many opponent boats as possible, and doing it as quickly as possible. Using this game as a test bed, we designed two types of experiments. The first experiment evolved the attacking team against unmoving defenders placed around the flag. We experimented with different team configurations and sizes for the defenders, while the attacking team consisted of 5 boats. For the second experiment we coevolved both the attackers and the defenders. In this case, the defenders could move around to attack and defend, and both teams had equal configurations. The goals of the defender were to defend the flag for the time limit, and destroy as many opponent boats as possible while keeping player boats alive.

A chromosome in the evolutionary representation consisted of evolved parameters used by the IM function to generate IM weights for each game entity every time step. A single chromosome contained the IMs parameters for an entire team. Each entity in the team has a set of genes (parameters) representing: each team mate, each opponent entity, the flag. Figure 1 depicts how the chromosome is used for the evolutionary experiment. The chromosome at the top of the Figure displays the set of parameters for entity 3, which is given the label f3 (friendly 3). The parameters are assigned as $f0 - f4$ for the friendly team entities, $e0 - e4$ the enemy team entities, and flg for the flag.

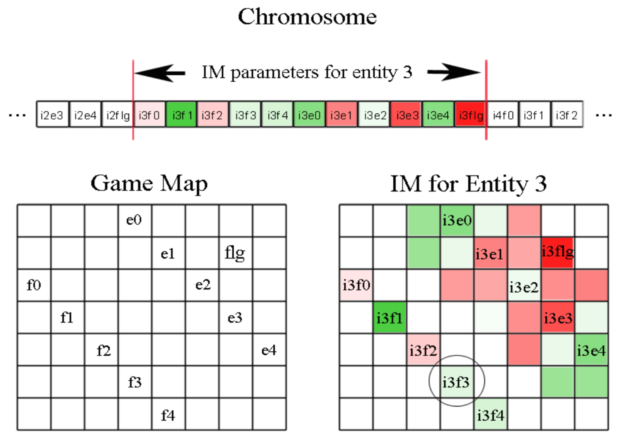

Chromosome
IM parameters for entity 3
... i2e3 i2e4 i2flg i3f0 i3f1 i3f2 i3f3 i3f4 i3e0 i3e1 i3e2 i3e3 i3e4 i3flg i4f0 i4f1 i4f2 ...
Game Map
e0
e1
flg
f0
e2
f1
e3
f2
e4
f3
f4
IM for Entity 3
i3e0
i3e1
i3flg
i3f0
i3e2
i3f1
i3e3
i3f2
i3e4
i3f3
i3f4

Fig. 1. The world map used

The evolved parameters are then used by the IM function to generate IMs for each entity in the team. The function is passed the parameters, along with the current game state map, and outputs the IMs for each entity in the team. This process is depicted in Figure 1, which outlines the generation of the IM for entity 3. The table on the left is the current state of the game map. This includes where in the world map all the entities are at the current time step. The IM function then takes the game map, and generates the IMs for all the entities represented in its team. The example IM generated for entity 3 is shown as the table on the right. This is created by using the parameters shown in the chromosome for entity 3, and applying these to the IM. The algorithm used for generating the IMs for the team of m entities is as follows:

```
for i in m
   for x in maxX
      for y in maxY
         if(not isOccupied(Map[x][y]))
            IM[i][x][y] = 0
         if (flag(Max[x][y]))
            genFlagInfluence(IM, i, x, y)
         if (friendly(Map[x][y])
            genFriendlyInfluence(IM, i, x, y,
            map[x][y])
         if (enemy(Map[x][y])
            genEnemyInfluence(IM, i, x, y,
            map[x][y])
```

`genFriendlyInfluence` directly assigns the IM cell at x, y the corresponding chromosome allele for the entity represented at Map[x][y], from the corresponding gene set for i. For example, referring to Figure 1 ($i = 3$), if $Map[x][y] == f1$, then the corresponding gene on the chromosome for entity 3, would be the value represented by the strong green colour (a repulsive influence). This value is then directly added to the IM cell at location x, y. The same form of assignment is done for the flag value with `genFlagInfluence`. `genEnemyInfluence` differs slightly in the assignment. It sets the evolved parameter value (v_i) at location x, y as per `genFriendlyInfluence`, as well as setting reducing values for all cells neighbouring location x, y in range r. The values are assigned as per below:

$$IM[i][x'][y'] += v_i - \frac{v_i}{r \times max(x', y')}$$

where x' and y' are the current row and column in range $(x, x+r]$ and $(y, y+r]$ respectively,v_i is the evolved parameter value for the enemy entity, and r is a fixed constant representing the enemy weapon's firing range (in experiments $r = 4$). Since it is possible for multiple entities to be in the same map/IM cell or close enough for their influences to overlap, the final IM cell value is the sum of all influences on that cell, hence the += notation in the formula above.

The generated IM is then used by an entity to assign a goal and movement towards the goal. To assign a goal, the most attracting weight on the IM is found (in the range [-50, 50], with -50 the most attracting, and 50 repulsing), and assigned as the goal. The A* path finding algorithm is then used to find a path to the goal, taking into account the attracting and repulsing influences of the IM. By using the A* algorithm, combined with the IM influences, a path can be created for the entity that avoids potential conflicts. Additionally, as the IM changes with the state of the game, the path can change dependent on the current influences. This can effectively change spatial tactics for the entity.

The previous experiments showed the evolutionary system to be an effective way to discover simple tactics given different scenarios. The IM representation showed potential for complex strategy creation. The evolution assigned weights that allowed entities to follow friends, attack enemies and capture the flag. Additionally, tactics evolved that assigned consecutive weights as goals. When the most attractive weight assigned a goal to a friendly or enemy boat, when the boat was no longer in play (were destroyed), a new goal was set. This meant that when a friend being followed or an enemy being attacked were no longer on the battle field, the entity was able to assign a new goal and change tactics. By assigning consecutive weight allocations, the evolution created tactics such as coordinated attacks on a group of enemy entities, followed by a charge on the flag when the game state changed.

The research was a first step in acheiving a larger goal. The representation needed further testing, for more complex scenarios such as harder game maps. Additionally, the co-evolution experiments did not yield overly positive results. The game scenario proved too simple and unbalanced to see interesting tactics develop. Additionally the coevolution found imbalance in the game design. The use of a team of defenders and a team of attackers created an asymmetric team design. This allowed the attackers to have an optimal tactic that the defenders were not able to overcome, which was for all entities to charge (rush) straight for the flag. We attempted to directly evolve against this tactic, but were not able to find any defender tactics that could counter the attackers. The next section describes the extensions we have performed to the previous research to address some of these issues.

III. Methodology

One of the main issues with the coevolution from previous research was that of game balance. To address this, we changed the game team structure. Previously, we found that the attacker and defender style of game, where one team defends a flag and the other team has to reach the flag, was difficult to balance. The defender team was unable to defend the flag against a tactic of all attacker units charging (rushing) the flag. For this experiment, we implemented a more symmetric scenario, where both teams were of equal strength, and had their own flag to defend. In this version, the first team to capture the flag within the given time frame won, otherwise a loss was recorded.

To implement this, we changed the chromosome to include two flags, as seen in Figure 2. Both populations of individuals used the same format of chromosome, with parameters for each entity (friendly and enemy) and each flag. By making the game symmetrical, we hoped to see more interesting tactics evolve than witnessed in the previous research. Additionally, we wanted to generate tactics that could beat the 'rush the flag' attacking strategy that was previously unbeatable. By using two populations, we wanted to seperate the two 'species' being evolved. In some earlier experiments we included teams that were not symmetrical, and this might be something we wish to investigate further in the future.

The next section discusses the implementation of the capture-the-flag game used.

A. *The capture-the-flag game*

This research used a simple game of capture-the-flag in an open water scenario, as seen in Figure 3. This game was chosen as it is a well known example of a simple game with complex tactics, and provides a good test-bed for our research. The open water scenario is an overly simple one, but for this stage of the research is sufficient to test the coevolutionary mechanism. In future work we will implement a terrain and provide a more complex scenario.

In this research there are two teams of players, each with five boats and a flag to defend. The goal is to capture the

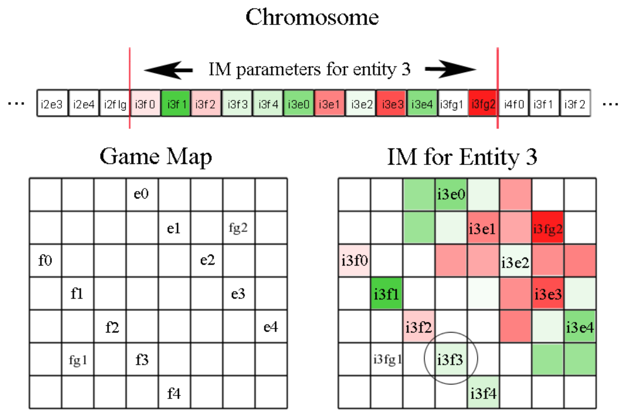


Fig. 2. Example chromosome section

opposition flag in the least time possible, with sub-goals to sink as many of the enemy on the way, while retaining as many own players as possible. The end of game scenarios are: either player team capturing the opposition flag and winning, or the game timer running out, in which case a time-out occurs and neither player wins. To capture the flag, a team has to hold the opposition flag (remain within the flag vicinity) for a short period of time.

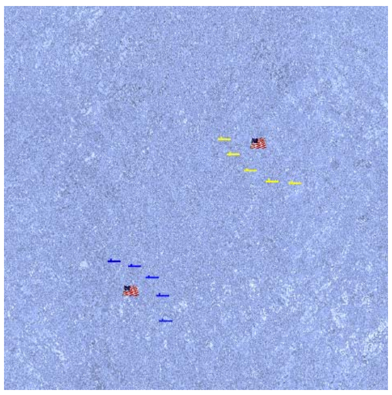


Fig. 3. The world map used

The mechanics of the game were kept very simple. The boats have an auto-firing mechanism that begins firing when an opposition boat is within a certain distance. If multiple boats enter a firing range at the same time, the closest boat gets targeted first. This firing mechanism is very simple, and not ideal. In the future, we will implement a more sophisticated method of when and whom to fire on. The game entities use boat physics which model a turning circle and acceleration/deceleration.

The next section discusses the coevolutionary system used to evolve tactics for the entities.

B. *The coevolution*

The coevolutionary system included two populations of individuals, with each species representing a different team in the game. The two teams started in the positions shown in Figure 3, one team at the top of the map, and one at the bottom. Each team had its own flag to defend, and an opposition flag to capture. The population of individuals starting at the top of the map were labelled population 1, and the individuals starting at the bottom were population 2.

The representation of the individual was as shown in Figure 2. The fitness is found by evaluating the individual against a number of individuals from the opposing population. The evaluation is to play a game, using the evolved IMs from the chosen individual to determine the movement of the entities in the game. The evaluation function is run on completion of a game, which occurs when one team captures the flag or time ran out. The results of the game are recorded, including the win/loss result, the number of player and opposition entities remaining, and the time remaining. These results are then used to record the fitness for the individual, with the following fitness function:

$$\begin{aligned} fitness = \; & winRatio \times winWeight \\ & + fWeight \times friendlyCount \\ & - eWeight \times enemyCount \\ & + time \\ & + win * oppWinRatio * oppWeight \end{aligned}$$

where $winRatio$ is the percentage of games won over all games played by the individual. The $winWeight$ then assigns the weight of the $winRatio$ on the fitness, and was set to 500. $friendlyCount$ is the number of friendly units left after the encounter. Similarly, $enemyCount$ is the number of enemies left. $time$ is the amount of time left if the individual won, with respect to the given time limit. For example, if the time limit was 60 seconds, and the individual reached the flag in 40 seconds, than $time$ would equal 20. The $oppWinRatio$ is the number of wins per games played for the opposition, which only came into play if win was 1 (True). The $oppWeight$ determines the weight the $oppWinRatio$ had on the fitness, and was set to 200. The use of the $oppWinRatio$ allows individuals that won against harder individuals to score a better fitness. This fitness is very rudimentary, and has a lot of room for improvement. One possibility is to investigate using a more multi-objective style, where individuals can be chosen for different areas of success.

Rank selection was used, which was deemed the most effective after experiments with rank, tournament and roulette selection. Elitism was applied to a percentage of the population, with the elite individual copied directly to the new population, and the variation operators applied to non-elite individuals. The variation operators used were two point crossover and mutation, where mutation occurred after crossover. We note however, that the two point crossover was chosen as a simple implementation solution. It is not the most efficient mechanism to use, particularly given the fixed mapping of the integer gene values, unlike binary encoding. In future research we would like to include a more sophisticated crossover mechanism that incorporates a disruption to the values surrounding the crossover points, allowing the crossover to become a search operator similar to binary two point crossover, similar to that described by Deb and Agrawal for real variable values [19]. The mutation combined small and large parameter perturbations with a smaller chance of a large mutation occurring. The small mutation was to add or subtract 1 from the parameter. Large mutation picked a new random integer between the -50 to 50 range of all parameters.

The next section provides the details of the experimental setting used, and the results. We describe the scenario and details for each experiment, and then give a summary of our observations and findings.

IV. Experiment and results

In order to efficiently obtain experimental results for the coevolutionary system, we implemented the described coevolutionary system on a parallel cluster. The evaluation for each individual was performed on separate nodes. A population size of 20 was used, and run for 100 generations. Initially 5 evaluation games were played for each individual against random unique individuals from the other population. This meant that each individual played at least 5 games, with a possible maximum of 20, due to random selection from the other population's evaluation. After initial successes, we then experimented with decreasing the number of games, which did not decrease the fitness results achieved.

The experiment parameters for the results described below are as follows. The crossover rate was set to 50% with a mutation rate of 1%. The elitism ratio was set to 5%. Each individual played a minimum of 3 games against unique individuals from the opposing population, with a possible maximum of 30 (all opposition individuals). The coevolution was run 10 times, with the best, worst and average fitness of the individuals for each generation recorded.

The average of all results for the best individuals at each generation, in both populations (overlaid), can be seen in Figure 4. Figure 5 then displays the average results of the runs for the maximum, minimum and average fitness of each generation for a population.

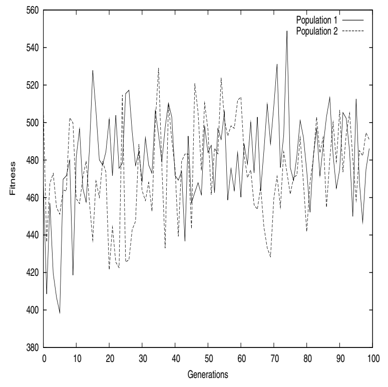
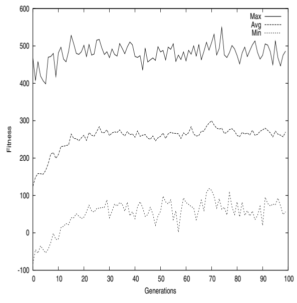
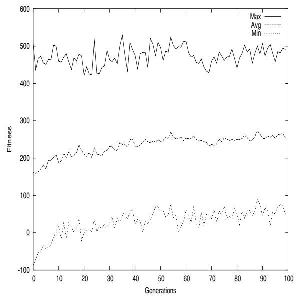


Fig. 4. Average of the best for both populations each generation

These results show a steep initial improvement, followed by a more gradual improvement over time. This was an improvement from previous experiments, which didn't see any improvement after the first 10-20 generations. We noticed that while the tactics being evolved tended to be either attacking or defensive, they were more adaptable. For example, while the opposition remained near the flag, the player boats would avoid attacking and remain in defensive mode. As soon as the opposition boats moved away from the flag or were destroyed, the player boats would switch tactics and rush the flag. This occurred due to the summation of weights used by the IM; if an enemy had a repulsing value, and the flag an attracting one, the repulsing vale of the enemy when on top of the flag would negate the flag's attracting value.



(a) Population 1



(b) Population 2

Fig. 5. Average best, minimum and maximum results of each generation in the populations

We were also observing the same 'following' techniques seen in previous research, where an entity's IM would assign the most attracting weight to a friendly entity and thus follow them. This continued to be used in strategies where an entity would follow a friendly entity until the friend was destroyed, and then switch tactics. The combination of this with the summation of weights described above resulted in some interesting and quite complex tactics. For example, a group of entities would all follow each other, grouping together to form a strong force for defensive/attacking purposes. When the opposition changed tactics, or was sufficiently weakened, the group would then split and serve different purposes, such as destroying single enemy entities, or rushing the flag.

Following our initial success, we performed experiments to test against the 'rush the flag' strategy discussed earlier. For these experiments, we implemented the strategy as a baseline measurement mechanism. Each generation, the best individual was played against the rush strategy. The average of the results from all the runs of population 2 against this tactic can be seen in Figure 6. The results show the fitness on the y axis as derived from the function described in section III-B. This showed the repeated success of the evolved strategies to generate solutions that could protect against the rush strategy, and even go on to win the game.

Further analysis showed that the evolved strategies that managed to beat the rush strategy were fairly similar. An example of one such strategy can be seen if Figure 7. The Figures show two screen shots from the strategy in action. The overlaid IM is for the entity at the bottom of the screen.

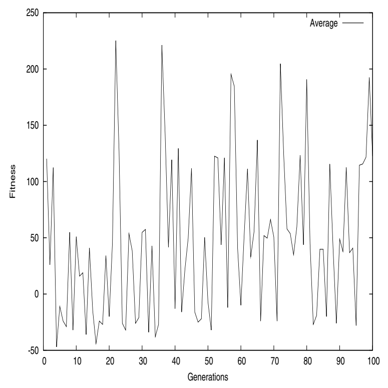


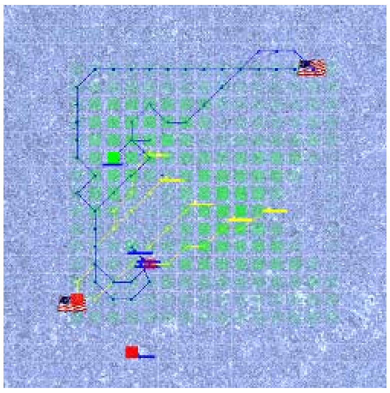
Fig. 6. Average of the best for both population 1 against 'rush the flag' strategy

As seen in Figure 7. (a), the majority of the entities in the team are moving together to attack the opposition. The entity at the bottom (with the highlighted IM) has its strongest weight representing itself, which means it stays where it is. In Figure 7. (b), we see that all but one enemy boat has been destroyed, and the two remaining team boats that were previously attacking have re-routed to the enemy flag. The one enemy remaining is heading for the flag, but it has a red (attracting) influence for the friendly entity at the bottom. When the influence of the enemy combines with the influence of the flag, the sum of the two becomes stronger than the attracting weight it has assigned to itself and the entity heads to the flag, destroying the last boat. The two other boats then go on to capture the flag.

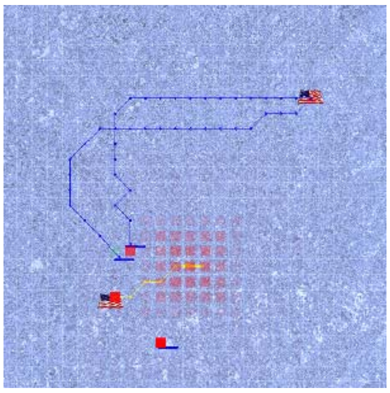
These results show that the coevolution was able to successfully find strategies that could overcome the rushing strategy from previous experiments. It was also able to do this repeatedly throughout the coevolution, even though it had no knowledge of, or incentive to beat the strategy. This demonstrated the success of our method for finding interesting strategies that could be generically applied.

To further test this, we implemented a defending strategy that was similar to the strategies we were seeing evolved. The strategy used four of the five boats to defend the flag, while the remaining boat headed for the enemy flag, avoiding enemy boats. The results against this strategy can be seen in Figure 8. The first thing to note, is that the runs were only able to briefly find a winning strategy one time, and were not as good against this strategy. This was somewhat expected, as this strategy is a much harder to achieve a win with. However, it is worth noting that although a win was rarely achieved, the evolutions gained a rapid improvement towards regularly achieving a draw. Given that it was also showing continued improvement, it is possible that further generations could have achieved a better result.

The final experiment we applied was to test the general effectiveness of evolved solutions. We picked the best individual found across all the runs, and then using that run, we selected the best individual from the opposition population. We then played each of these individuals against the best



(a) Stage 1



(b) Stage 2

Fig. 7. Example evolved strategy vs. the attacking strategy

idividuals from their opposing population in all the other runs. This shows us how well the best individual fared against other runs, and how well the best opponent it was directly evolved against fared.

The best was chosen as the individual that achieved the highest fitness plus the highest score against the 'rush the flag' strategy. The results from the two chosen individuals can be seen in Figure 9. Figure 9. (a) shows the results for the best individual, while Figure 9. (b) shows the results for the opposition individual. The results show that the best individual stands up well to earlier generations of all runs, but in comparison to the results for the rushing strategy, it is a much easier strategy to beat. The opposition individual was even easier to beat, and from an earlier generation. This shows that there are still improvements we can make in our coevolutionary method. We are hoping that we can achieve stronger general strategies through further research. In future work we would like to make changes to the variation operators, and leave the coevolution running for longer with a larger population size, observing if this helps the matter.

Finally, when analysing the evolved strategies, they were still comparatively inflexible when compared with human players. The individuals have a tendency to get 'stuck', when they have obtained all their goals, and no longer adapt to the scenario except where the IM changes. This can be

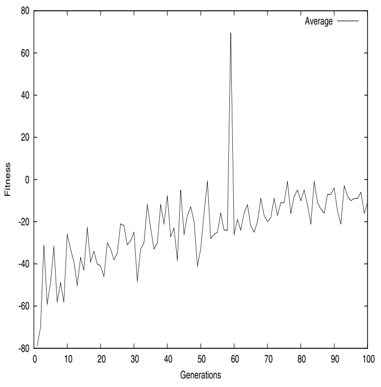


Fig. 8. Average of the best for both population 1 against a defending strategy

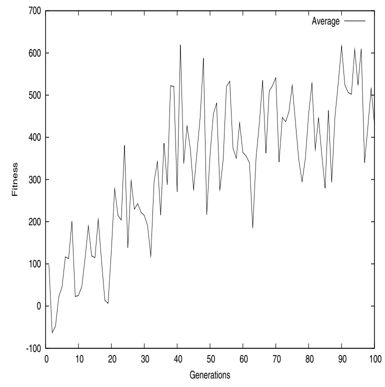
detrimental if there is only one or two entities left, and none have the opposition flag as a goal, causing a stale-mate. It may be possible to hard code some end game tactics into the individuals, but ideally we would like this to be self-discovered.

V. Conclusions and future work

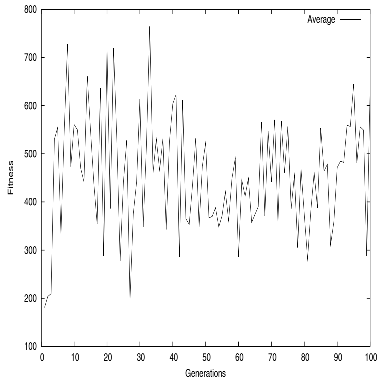
We have shown that our method of coevolving Influence Maps can be successfully applied to generate coordinated tactics. The coevolution was able to generate tactics that changed with the scenario, and provided general solutions that show promise for the technique.

Following on from this research, we would like to further change the way our game is represented. Our first step is to include land mass into the game. The addition of land mass adds new complexity to the game, and allows further opportunity for complex strategy generation. Additionally, we would like to implement more complex firing and movement strategies. This would include giving the evolution additional information regarding speed, and possibly health. For speed we are considering including another layer of influence maps that provide the speed ranges an entity should achieve at spatial points on the map. These points could represent areas surrounding land and other entities. This allows the entity to make decisions on how fast/slow it wishes to go in relation to other entities on the map. By controlling speed, the evolution should be able to use this to generate more complex tactics, such as more complex feinting, and surprise attacks. The addition of health information could be applied to the influence range for enemy entities. This should allow for creation of more complex sneak and avoidance/attacking measures. To do this, we can change the IM function so it uses an evolved range, determined by the health for each enemy entity. Currently this is set at a range of 4. Using the health to determine the range will determine the range of influence for each entity. We could then use this range to determine when an entity should start firing on an opponent.

We would also like to extend the current coevolution implementation. As mentioned, we want to experiment with different forms of crossover, and find one that is more suited



(a) Population 2 runs vs. individual from population 1



(b) Population 1 runs vs. individual from population 2

Fig. 9. Playing each population against a winning strategy evolved in the opposing population

to our problem. We would like to implement experiments with the new crossover mechanism, and expand the diversity of the population for longer. By implementing a longer search with greater diversity, we hope to achieve more complex strategy generation. Additionally, we will include a form of memory in the coevolution, where older evolutionary strategies can be used to guide evolution of new and more complex strategies. We would also like to experiment and compare other forms of representation. While the evolution of IMs have shown great promise, they may not prove general enough in more complex games. It may be possible to include a hybrid of different techniqes that would prove more beneficial.

Finally, our end goal is to create an opposition player that can adapt to a human player in between games. To achieve this goal, we intent to create a more complex game environment with more RTS game mechanisms. We then hope to apply the techniques mentioned in this research, and provide an opposition that works autonomously in a coordinated manner, and can adapt to the human player. The adaption will require the human to play against a coevolved individual, and a model of the human player created using the game data and results. This model will then be used to steer the coevolution, and hopefully allow the adaption to the human strategy. In doing this, we will create a more challenging and interesting opponent for ongoing game play.

VI. Acknowledgments

This research is supported by ONR grants N00014-08-1-0781 and N00014-09-1-1121.

References

[1] C. Miles, "Co-evolving real-time strategy game players," Ph.D. dissertation, 2007.

[2] A. L. Zobrist, "A model of visual organization for the game of GO," in *AFIPS Conf. Proc.*, 1969, pp. 34, 103–112.

[3] P. Tozour, "Influence mapping," in *Game Programming Gems 2*, M. DeLoura, Ed. Charles River Media, 2001, pp. 287–297.

[4] N. Wirth and M. Gallagher, "An influence map model for playing Ms. Pac-Man," in *Symposium on Computational Intelligence in Games*. IEEE Press, 2008.

[5] C. Miles and S. J. Louis, "Towards the co-evolution of influence map tree based strategy game players," in *Proceedings of the 2006 IEEE Symposium on Computational Intelligence in Games*. IEEE Press, 2006, pp. 75–82.

[6] ——, "Co-evolving influence map tree based strategy game players," in *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Games*. IEEE Press, 2007.

[7] ——, "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.

[8] S. J. Louis and C. Miles, "Playing to learn: Case-injected genetic algorithms for learning to play computer games," *IEEE Transactions on Evolutionary Computation*, pp. 669–681, 2005.

[9] M. Bergsma and P. Spronck, "Adaptive spatial reasoning for turn-based strategy games," in *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*. AAAI, 2008.

[10] D. B. Fogel, *Blondie24: Playing at the Edge of AI*. San Francisco, CA: Morgan Kaufmann, 2002.

[11] S. Chong, D. Ku, H. Lim, M. Tan, and J. White, "Evolved neural networks learning othello strategies," in *The 2003 Congress on Evolutionary Computation*, vol. 3, Newport Beach, California, USA, 2003, pp. 2222 – 2229.

[12] P. Avery, G. Greenwood, and Z. Michalewicz, "Coevolving strategic intelligence," in *Proceedings for IEEE Congress on Evolutionary Computation*, Hong Kong, 2008.

[13] C. Miles, "Lagoon craft," 2007.

[14] P. Sweetser and J. Wiles, "Combining influence maps and cellular automata for reactive game agents," in *6th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2005)*, 2005, pp. 524–531.

[15] S. Jang and S. Cho, "Evolving neural NPCs with layered influence map in the real-time simulation game conqueror," in *Proceedings of the 2008 IEEE Symposium on Computational Intelligence in Games*. IEEE Press, 2008.

[16] P. Zigoris, J. Siu, O. Wang, and A. Hayes, "Balancing automated behavior and human control in multi-agent systems: a case study in RoboFlag," in *American Control Conference*, vol. 1, 2003, pp. 667–671.

[17] H. Danielsiek, R. Ster, A. Thom, N. Beume, B. Naujoks, and M. Preuss, "Intelligent moving of groups in real-time strategy games," in *Proceedings of the 2008 IEEE Symposium on Computational Intelligence in Games*, 2008, pp. 63–70.

[18] P. M. Avery, S. J. Louis, and B. Avery, "Evolving coordinated spatial tactics for autonomous agents using influence maps," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Games*. IEEE Press, 2009.

[19] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," in *Complex Systems*, vol. 9, 1995, pp. 115–148.