# Evolving Spatial Tactics using Influence Maps

Phillipa Avery, Sushil J. Louis and Benjamin Avery

*Abstract*— **We evolve tactical control for entity groups in a naval real-time strategy game. Since tactical maneuvering involves spatial reasoning, our evolutionary algorithm evolves a set of influence maps that help specify an entity's spatial objectives. The entity then uses the A* route finding algorithm to generate waypoints according to the influcene map, and follows them to achieve spatial objectives. Using this representation, our evolutionary algorithm quickly evolves increasingly better capture-the-flag tactics on three increasingly difficult maps. These preliminary results indicate (1) the usefulness of our particular influence map encoding for representing spatially resolved tactics and (2) the potential for using co-evolution to generate increasingly complex and competent tactics in our game. More generally, this work represents another step in our ongoing effort to investigate the co-evolution of competent game players in a real-time, continuous, environment that does not assume complete knowledge of the game state.**

## I. Introduction

Evolving group tactics in our research involves using spatial information to determine relative entity positions and then maneuvering to achieve a set of spatially resolved objectives. We are specially interested in generating new and interesting tactics and in player modeling. That is we are interested in methods of generating tactics that we and our subject matter experts may not have come up and we are interested in recording and learning from players' game play. As a first step in our overall research objectives, this paper investigates the evolution of boat group tactics in a naval real-time capture-the-flag game.

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 co-evolving real-time strategy game players pointed out the importance of tactical superiority when strategies and resources are balanced among competing players [1]. Real-time stategy games emphasize "strategy," but achieving strategic objectives depends on tactics. Bad tactical planning, an inability to adapt quickly, and unskilled maneuvering can ruin good strategy. Tactical planning thus plays an important role in many situations.

Although we have discussed tactics and strategy, we have not yet defined these terms. For our work, we find it useful to define strategic planning as planning that uses all available resources to achieve a goal such as winning the game. Tactical planning uses a subset of available resources to achieve an objective that supports the over-arcing strategic objective of winning the game. Tactical and strategic planning are the same when there is no grouping (no resource subsets) in a game.

Phillipa Avery and Sushil J. Louis are with the Evolutionary Computing Systems Laboratory (ecsl.cse.unr.edu), Dept. of Computer Science and Engineering, University of Nevada, Reno, Nevada, USA; (email: pippa, sushil@cse.unr.edu).

Hierarchical tactical control approaches have the advantage of a central intelligence guiding group behavior. Classical knowledge-based techniques are particularly well suited to centralized intelligence approaches. However, losing the leader (the central intelligence) can lead to confusion and significantly degrade tactical adaptiveness. The obvious counter tactic spends resources in finding and destroying the leader then mopping up the disorganized remaining enemy forces. An interesting question here is: How can we identify the leader of a group, especially when we have incomplete knowledge of the game-state? We do not address this issue here.

Distributed tactical control approaches represent the other end of the spectrum. Because of the non-linear nature of interactions among independent entities trying to act together to achieve an objective, writing a good distributed controller is non-trivial. Evolutionary computing approaches such as genetic algorithms were designed to attack such non-linear problems and as such seem particularly suited to generating distributed tactical controllers. We believe computational intelligence approaches therefore represent a good approach to designing automated, adaptive, tactical controllers. In this paper, we use evolutionary algorithms to try evolve tactical control for small groups of boats as we work towards tactics that involve deception, sacrifice, feints, flanking, splitting enemy firepower, and hopefully others that will surprise us. We want the game entities (boats) to function autonomously, but still work with the rest of their group in achieving an objective. The autonomous behaviour allows the boat some independence - to move and attack on its own, while the coordination provides an overall tactic for the attacks.

To achieve the necessary level of autonomoity and coordination, we have designed a representation based on Influence Maps (IMs). An evolutionary algorithm evolves a set of parameters that define a set of IMs, one IM for each boat in the group. This mechanism allows coordinated attacks by providing the same influence values for attacks points on the map. Individually, the IM for a boat gives direction and goals for the boat independent of other boats in the fleet. Cooperatively, by evolving the IMs together as one individual in the population, the IMs work together to obtain the overall reward given by the evaluation function. Our results show that this representation can lead to competent tactical behavior including the generation of sub-objectives that represent sequential steps towards an overall tactical objective. We believe the early results presented in this paper indicate the potential of our approach for generating competent tactical behavior for entity groups.

This paper is organized as follows. The background section gives a brief description of IMs and the other relevant work done in this area. The methodology section describes the mechanisms used in the research, and the setup for the experiments performed. The results section provides the results from the experiments, along with a discussion on the observations made. The conclusion provides a summary of the research presented, along with a plan for the future work still to be done.

## II. Background

An influence map is a grid placed over the game map with values assigned to each grid cell by some function. Once computed, an influence map can encode spatial features or some spatially related concept. They can identify boundaries of control, "choke points", and other tactically interesting spatial features. It is interesting to note that influence maps evolved out of work done on spatial reasoning within the game of Go and have since then been occasionally used in video games and the AI and CI in games communities [2], [3], [4], [5], [6], [7], [1], [8].

Prior work in our lab evolved a single influence map **tree** to generate objectives for the LagoonCraft RTS game [9]. These objectives were then achieved by a genetic algorithm generated near-optimal allocation of resources [1]. The current work differs in that we are evolving cooperative influence maps for a group of entities that coordinate to complete a task. Each entity uses an individually assigned influence map, 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.

Bergsma and Spronck also use influence maps to generate tactics for a turn based strategy game [8]. A neural network layers several influence maps to generate attack and move values for each cell. Like Miles' work, Bergsma and Spronck combine influence maps to generate tactical or strategic objectives. Unlike Miles' work, they use neural networks (whose weights are evolved) to combine the influence maps.

To the best of our knowledge, this is the first attempt to evolve and 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.

The prior work cited in this section seems most relevant, but there is much other work in using influence maps for spatial reasoning, organizational behavior, and in other non game related fields [**?**], [**?**], [**?**]. We next describe the game, our representation for tactics in the game, and our fitness computation.

## III. Methodology

This section describes the evolutionary algorithm and representation used to generate spatial tactics. Before doing so however, we provide the game context within which we work.

### A. The Game

We implemented a simple game of capture the flag. Figure 1 depicts this scenario which has two teams of boats. Attackers in green occupy the lower portion of the figure while stationary defenders defend the flag on the upper right. The attacking team's goal is to reach the flag positioned behind defenders within a limited time. Defenders, in turn, must stop the attackers from reaching the flag in the given time limit.

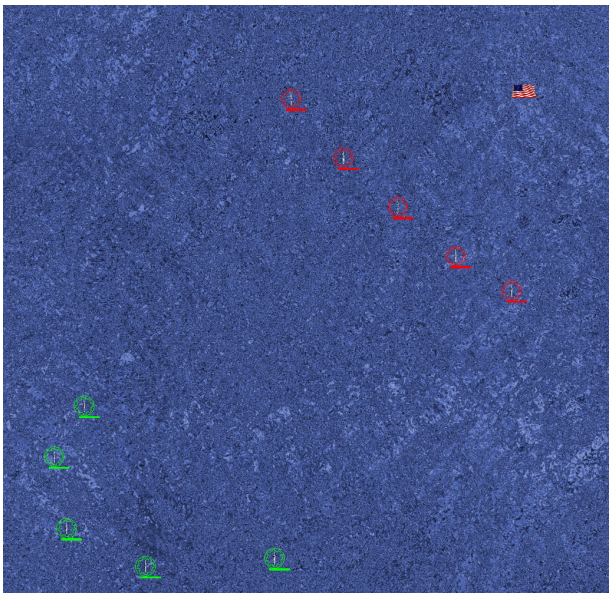

Fig. 1. Capture the flag game scenario

In this simplified game, defenders maintain their starting position and only fire when an attacker is within firing distance. The evolutionary algorithm evolves attackers. The goal is to capture the flag in the least time possible, and sink as many of the enemy on the way while retaining as many of the attackers as possible. Next, we move to a co-evolutionary model where we evolve both attackers and actively maneuvering defenders. The co-evolutionary version has the same goal for the attackers, with the goal of the defenders to stop the enemy reaching the flag while sinking attackers and retaining defenders.

The game entities use simple boat physics which model a turning circle and acceleration/deceleration. This game is continuos. Weapons fire automatically, with a firing range set for each boat. As soon as a boat enters the limited firing range, its weapon fire. If two boats enter the firing range at the same time, the closest boat gets targetted.

### B. Evolving Influence Maps

Each entity has its own influence map and an entity moves towards the current **lowest** value on its IM. The location of each entity in the world is marked on the IM with a value, $v_i$, and a radius of influence, $r$, corresponding to the firing range of the enemy weapon. For each entity, our evolutionary algorithm evolves the $v_i$ value for all entities in

the world. The radius of influence is fixed. Thus if there are 5 attackers, 5 defenders, and one flag, we have a total of 11 game entities. Given that we are evolving 5 IMs for the 5 attackers, we will have 11 values to evolve ($v_1$ to $v_{11}$ ) for each of these 5 attackers for a total of $11 \times 5 = 55$ values (or parameters). Thus each individual in the population consists of a 55 parameter length string.

In general, given $n$ game entities (including the flag) and $m$ of these entities to be controlled by evolved IMs, the evolutionary algorithm searches through the space of $n \times m$ parameter values to generate $m$ IMs that control our group of $m$ entities in carrying out a tactical capture-the-flag mission.

Here's the algorithm for generating the IM for one entity.

```
for x in maxX
  for y in maxY
    if(not isOccupied(Map[x][y]))
      IM[x][y] = 0
    if (friendly(Map[x][y])
      genFriendlyInfluence(IM, x, y, friendID)
    if (enemy(Map[x][y])
      genEnemyInfluence(IM, x, y, enemyID)
```

`genFriendlyInfluence` sets the IM cell value at location $x, y$ to be the evolved parameter value ($v_i$) for the entity ($i$) at that location. `genEnemyInfluence` does a little bit more. In addition to setting the IM cell value at location $x, y$ to be the evolved parameter value for the enemy entity at that location, the IM cell values for cells neighboring the location $x, y$ get reduced according to the following formula:

$$IM[x'][y'] += v_i - d \times \frac{v_i}{r}$$

where $i$ is the enemy entity id for enemy entity $e_i$, $v_i$ is the evolved parameter value for $e_i$, $d$ which ranges from 1 through $r$, is the current cell's distance from $e_i$'s location and $r = 4$ is a fixed constant representing the enemy weapon's firing range. Since it is possible for multiple entities to be in the same map/IM cell or close enough for their influences to overlap, an IM cell value is the sum of all influences on that cell, hence the += C-programming language notation in the formula above.

The EA uses a simple two point crossover, rank selection, and mutation. The mutation combines 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 involved picking a new random integer between the $-50$ to 50 range of all parameters. The evaluation function was run after an individual had completed a game, that is, when time ran out or the team captured the flag.

The evaluation function was:

$$\begin{aligned} fitness &= winScore + fWeight \times friendlyCount \\ &\quad -eWeight \times enemyCount + time \end{aligned}$$

where $winScore$ was 1000 if the flag was captured and 0 otherwise. $friendlyCount$ was the number of friendly units left after the encounter. Similarly, $enemyCount$ was the number of enemies left. $time$ was the amount of time left with respect to the given time limit. For example, if the time limit was 60 seconds, and the individual reached the flag in 40 second, than $time$ would equal 20. An additional reward was given if all the enemy boats in the scenario were sunk. The fitness function was then a maximizing function. It should be noted that this evaluation function is a very simple start, and can be improved upon. This is something we will be investigating in future research but for now, this simple function enabled us to get to the results described in the next section.

### C. Using the Influence Maps

To use the generated IM, the boats use the A* algorithm to generate a path. A* uses IM cell values to plot a path from the entity's current location to the minimum value on this entity's IM. The lower the IM cell value ($-50$ to 50), the higher the **chance** of creating a path to that object. A* runs every time the destination location (the lowest IM cell value location) changes.

This destination can change if: 1) the destination is no longer in the game (e.g. a boat sinks), and 2) the boat moves. If one friendly entity's A* destination location is another friendly entity (a teammate), we will see friendly boats following each other. Given an admissable heuristic, A* is optimal but computationally expensive. To speed up our runs, we did not run A* every time the destination location changed but only when the destination location changed by over 300 units.

Using such an IM combined with A* for movement for each boat allows coordinated attacks when assigning similar weights for the enemy boats and allows attackers to follow each other as described above. The IM can also encode for a sequence of destinations (objectives). For example, if the miminal IM value was for an enemy entity and the next lowest value was for the flag, then the boat using this IM would attack the enemy entity and *then* move towards the flag (the next objective) when the enemy entity gets destroyed.

This ability to generate a sequence of objectives can lead to more complex tactics. For example, the assigned IMs could draw fire to some boats letting other boats move past to take the flag, or hold back some boats waiting for other boats to engage the enemy and then dart past once the battle has begun.

Finally, note that as the game state changes, the IMs for each boat in the attacking group changes.

We began our evaluation with three simple scenarios where stationary defenders protect a flag. Next, we try to model a somewhat more hierarchical tactical controller by having a leader-entity whose destruction can reduce tactical effectiveness. Finally, we look at co-evolving attackers and defenders.

### D. Co-evolution

We changed the game to include an active defence of the flag defenders. This means that defenders needed their own IMs to specify defensive tactics and we co-evolved a population of attackers against a population of defenders. In

this version of the game, attackers won if they captured the flag. Defenders won by destroying all attackers or if they succeeded in fending off attackers from capturing the flag within the game-time limit. The new fitness was calculated as follows:

$$\begin{aligned} fitness &= winRatio \times winWeight \\ &\quad + fWeight \times friendlyCount \\ &\quad - eWeight \times enemyCount \\ &\quad + time \end{aligned}$$

where $winRatio$ was the percentage of games won over all games played by the individual. The $winWeight$ then assigned the weight that the $winRatio$ has on the fitness. The rest of the function variables are the the same as before. We computed an individual's $winRatio$ by playing every individual in the population against five randomly chosen opponent individuals. Each individual could therefore actually participate in more than five games per generation with a possible (but very unlikely) maximum of $popSize \times 5 + 5$.

## IV. Results

This section describes the experimental scenarios that we implemented in more detail and discusses the results obtained on these scenarios. We begin with a simple experiment to test the efficacy of our representation, followed by increasingly difficult scenarios.

### A. Scenario 1

The first experiment we performed was using the simple scenario as depicted in Figure 1. This scenario is very simple to find tactics for, and was simply intended to test the viability of our encoding (the IM).

The experimental parameters are as follows. We used a crossover rate of $30\%$ with a $10\%$ chance of mutation. If it was time to mutate, then there was a $10\%$ chance of large mutation, otherwise our small mutation operator was applied. We used some elitism with the top $10\%$ of the population copied to the next generation. For this experiment, the population size was $10$. We manually stopped evolution as there was no set runtime. We used a small population size for testing and because each evaluation takes a long time. During fitness evaluation, each game ran for a maximum of $50$ seconds. $fWeight$ and $eWeight$ were both $12$. The $winScore$ parameter was assigned $1000$ if the individual won, and $0$ otherwise.

We repeated the above experiment ten times. It became apparent that almost every time, there was at least one individual in the initial randomly assigned population that had a winning strategy. This turned out to be a very simple scenario. Additionally, all the runs evolved what we would consider a 'good' tactic for this scenario by the end of the $4^{th}$ generation. An example of the tactics evolved can be seen in Figure 2.

The Figure shows the friendly boats indicated by by green circles, and the enemy boats indicated by red circles. The flag can be seen in the top right corner. The bottom right friendly boat has been selected (seen by the solid coloured selection circle), and the IM for the boat is shown on the grid. The IM weights are depicted with green as a positive value (avoidance point) and red as a negative value (attraction point). The strength of the weight is represented by the transparency of the colour. For example, a highly negative value (-50) will produce a strong green color that fades as the value reaches closer to zero. The A* path for each friendly boat is also depicted by the green lines going from the friendly unit to their destination.

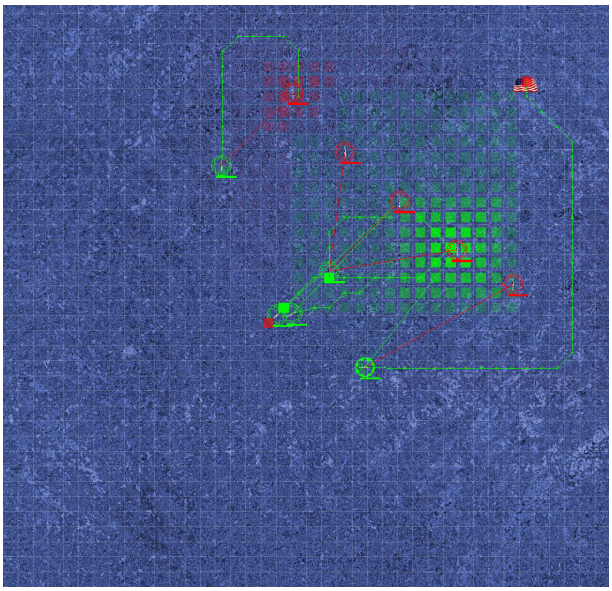

Fig. 2. Example tactic for scenario 1

In the case of the selected unit in this example, it can be seen that the strongest negative value, and hence the A* path destination, is assigned to the flag. The path skirts around the strong green negating values assigned to one of the enemy boats, and heads up to the flag. While the boat is fired upon, it survives to make it through to the flag. Other tactics similar to this were evolved, along with tactics that set a straight path through to the flag. As the boats can survive long enough to 'dart past' the defenders, they make it to the flag quickly, saving time.

### B. Scenario 2

To counter the ability to skirt around the edges, and to increase the damage taken when heading straight past defenders, we changed the scenario to that depicted in Figure 3. We used the same experimental parameters as in Scenario 1. The initial generation very rarely succeeded in capturing the flag. Instead, over time, evolution succeeded in generating more complex successful tactics. To counter the defense in this scenario, the entities developed coordinating tactics (IMs), where pairs or groups of boats would work together to attack defensive areas. One way was to have one boat follow another and have the follower make it through to the flag after the followed boat drew enemy fire and sank. Another tactic was to have two boats attack the same enemy boat. You can

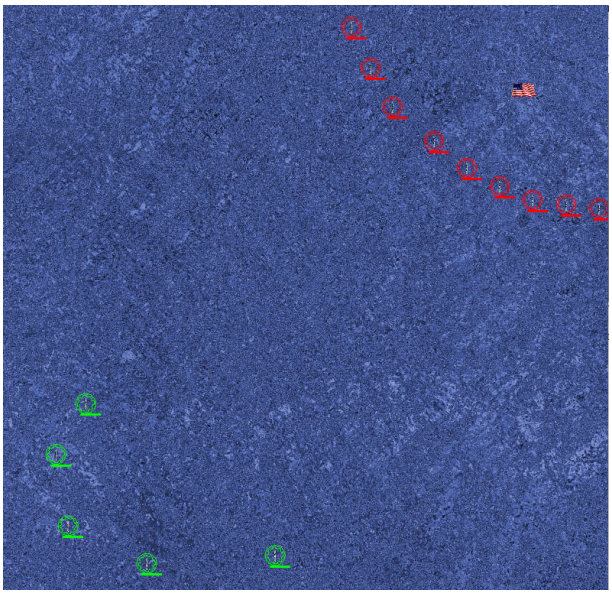

Fig. 3. Setup for scenario 2

see an example in Figure 4.

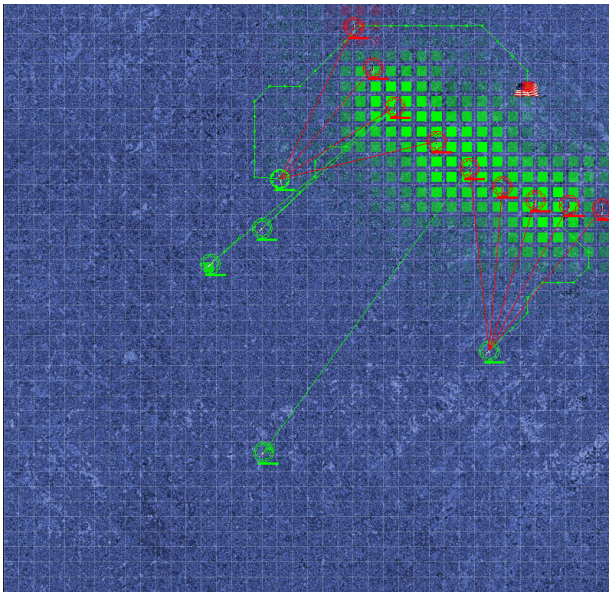

Fig. 4. Example tactic for scenario 2

This example has the top-most friendly unit selected, hence the IM being viewed is for that boat. As can be seen, the flag is the strongest red value and set as the A* destination. On the way to the goal, the first defender (red) boat must be passed. By creating a path that travels through the enemy's flank, the enemy firing rate is reduced, as the number of enemies firing at the boat is only two (the second defender in the line still in range), as opposed to three or more at any other location. Even with only two boats firing however, the boat would be sunk before reaching the flag. To counter this cooperative tactics are used where the two friendly units below the selected unit will reach the third enemy boat first. The firing focus of the second and third (and fourth) enemies will be on these boats, allowing the selected boat to move past taking fire from the first defender boat only, hence making it to the flag.

Other tactics included sending boats out in groups of two or more following the same path. The first boat would take the focus of the firing, while the second boat would dart past to take the flag. One common factor was that to win in this scenario, the IMs evolved needed to use some form of coordinated group attack.

These results strongly indicate the suitability of our IM-based representation for evolving competent group tactics. To evaluated the generalizability of these initial results, we designed another scenario: Scenario 3.

### C. Scenario 3

The most difficult scenario we tested for this research was a 13 boat defense as depicted in Figure 5. This scenario was designed (by us) to stop the "dart" tactics evolved. By including two rows of defenders, any attackers coming past the first row would be sunk by the second. Due to the increased complexity, we increased the population size to 20. As there were large numbers of enemies in the scenario, we also increased $eWeight$ to 15, and decreased $fWeight$ to 5. After watching a few generations of the first run, we

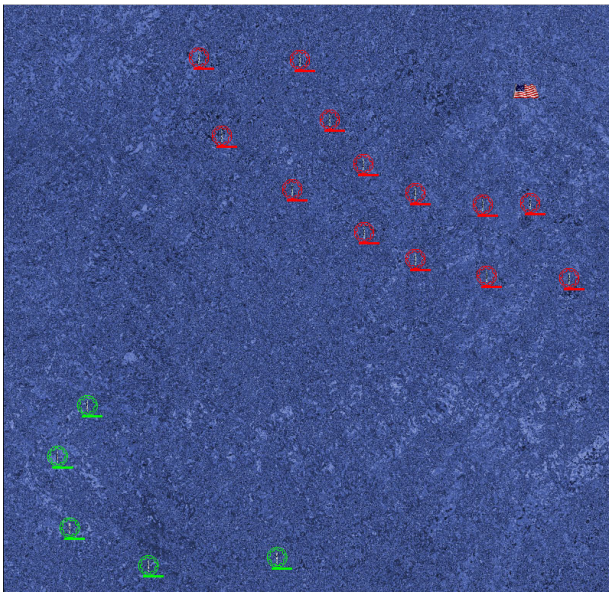

Fig. 5. Setup for scenario 3

found that individuals were not performing very well, and were doubtful about the creation of good tactics. However, at around 25 generations, we were pleasantly surprised to find good coordinating tactics emerging. This behavior was consistent across multiple runs. One of these tactics is depicted in Figures 6 and 7.

Figure 6 shows the IM for the leading attacker boat. It shows that the IM given to the boat assigns a high priority

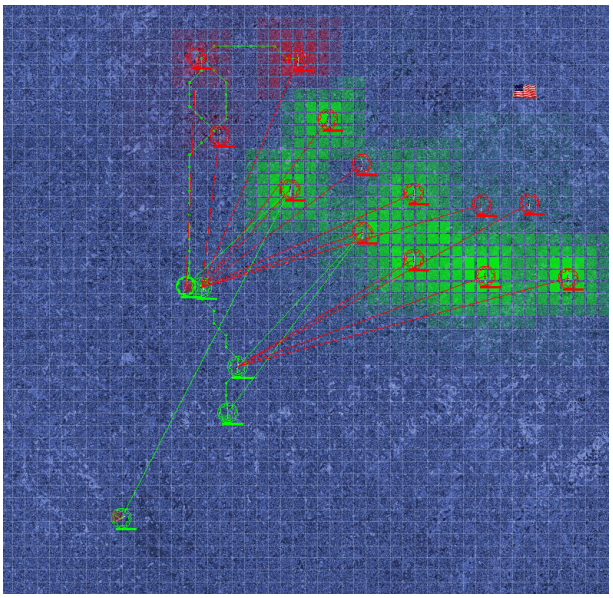

Fig. 6. Example of tactic for scenario 3 - stage 1

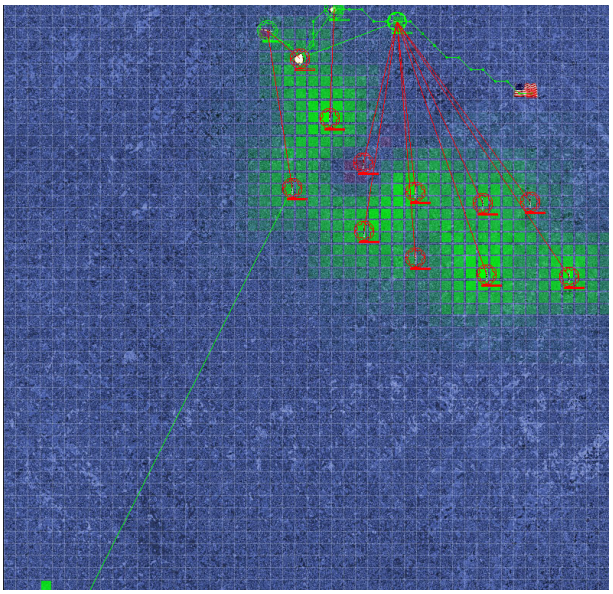

Fig. 7. Example of tactic for scenario 3 - stage 2

to attacking the top three defenders, with the flag having a slightly lower priority. Other boats in the fleet had a similar IM. Figure 7 shows the same game at a later stage and with a different boat's IM. Here we can see that two of the defending boats have been sunk, with a third close to being sunk (as indicated by the boat's health bar). The IM also shows that once the first defending boat died, our attacker switched its A* destination to the flag, and a new path was created to the flag that skirted the remaining boat in the way. The result of this tactic was that three enemy boats were sunk, four attackers survived, and attacker captured the flag. The fourth attacker remained at the bottom of the screen, with a lowest IM weight on itself. This likely occurred as it was not needed to win, and the bonus given for its survival in the evaluation function rewarded it staying out of the battle. This could have potential in evolving leadership behavior.

Figure 8 graphs average best fitness over time for ten runs on a scenario with nine defenders in two rows. After an

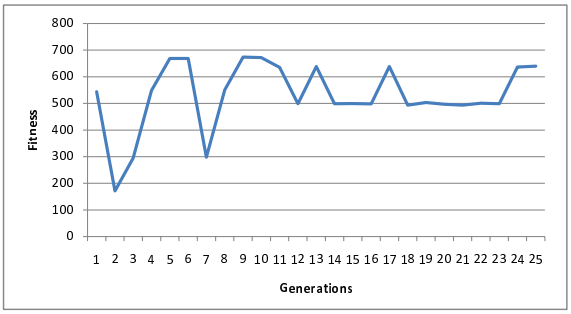


Fig. 8. Average Fitness

initial jump from the randomly generated population, fitness climbs to a relatively steady level by about generation 12. This seems to be representative of our runs on the scenarios described above.

### *D. Including a leader*

We initially designate one of the attacker's boats as a leader. If the designated leader sinks, no new A* paths are generated for the remaining attackers and they cannot change their objectives (tactic). We chose this approach over ending the game if the leader sinks, as we wanted the game to continue and see if the remaining attackers capture the flag anyway. We think of this as a very simplified model of tactical planning in the face of command loss.

After performing a few runs using Scenario 3 with the first attacker designated the leader, it became obvious this approach was not working exactly as planned.

Rewrite: [1]

### *E. Co-evolving attackers and defenders*

We setup up scenario 1 again except that now defenders could move. Several implementation issues with A* needed to be resolved. For example: both sides now needed to re-compute each A* path every time the destination point moved, and the only stationary point was now the flag. This additional A* computation affected the realism of movement for each boat and we had to spend some time tuning game parameters to our satisfaction.

Having two populations of individuals and each individual playing five games, a run now took much longer. We also had to increase the game time limit since attackers needed to

[1]The winning tactics were using all the boats to create a hole in the defense. Once the leader was killed, the remaining boats floated past (as their A* path was finished and not reallocated), and through momentum slowly made their way to the flag. While it is an interesting tactic, we're not sure that 'sacrificing the leader' would be a tactic the Navy would consider too realistic. What about the enemy with suicide boats? This is an area of research that we would like to continue in the future.

maneuver more around moving defenders to get to the flag. We increased the game-time limit to 90 seconds from 50. All this additional computation reduced the number of runs we were able to perform.

Even with these difficulties, it became apparent that after a few generations the defenders were generating two generic types of tactics. The first was a all out attack where defenders would head out and actively seek to destroy attackers. The second strategy was more balanced, a combination of attack and defense. Two or three of the defenders would circle the flag, while the remaining defenders would head out to destroy attackers.

In our limited runs to date, we have not observed (induced) the tactic(s) being used by the attacking team. Mostly they seem to spread out and try to sink the enemy, occasionally finding a path to the flag. We believe larger population sizes and longer run times are needed to get a co-evolutionary arms race that generates tactics, counter-tactics, and counter-counter-tactics to develop. We plan to parallelize our code and run on our cluster to see this and to test the limits of our representation.

## V. Conclusions and Future Work

From our experiments we were able to determine that the mechanism of simultaneously evolving autonomous but cooperative IMs was a success in the creation of group tactics. This is significantly different from other work in using IMs for planning [1]. Our purpose in this paper was to find a way for a group of boats to develop their own spatially resolved group tactics. Results show that we can simultaneosly evolve a set of cooperative IMs that entities can use to generate competent, adaptive, tactical plans.

Our initial experiment showed the potential for the IM mechanism when the randomly initialized first generation [2] to the scenario. This was due to the small number of objects on the game map, as the chances of finding a solution where one of the five boats made its way past the defense to the flag was fairly high. As the generations continued, more sophisticated tactics evolved and some simple forms of group-tactical planning emerged.

The second scenario encouraged more complex tactical planning by degrading the ability of an attacker to easily "dart" past defenders. The EA evolved effective group tactics that got around our new defensive posture. The attacking boats developed attack-and-distract tactics as well as flanking tactics to attack a single defensive emplacement in order to blast a hole in the defense that other attackers could exploit to get to the flag.

The third scenario proved more difficult. Eventually though, tactical solutions were found that involved objective chaining encoded by differing values of entities in the IM. These tactics involved chaining sub-goals together to finally capture the flag. This usually involved an initial goal of all boats attacking an area of the defense, then when done (defender sunk), going on to the next lowest value in the IM representing capturing the flag.

This ability of the IMs to change the tactics as the game state changes was something we were hoping to see, but uncertain if it would actually work. The evolutionary runs for the third scenario demonstrated effectively that it did indeed do as we had intended. The potential for this adaptive tactic representation is substantial, with applications for real time strategy and turn based strategy games. Theoretically, any situation where autonomous spatial tactics are required could benefit from the technique. For our purpose however, we are looking to incorporate the mechanism into more complex and realistic scenarios.

To accomplish our goals, we will need to extend this research to include the ability to react to harder scenarios. For example, once land mass is added to the scenario, how will this effect the tactics being created. Is it possible (and effective) to include the landmass in the IM as another evolutionary parameter, with the goal of depicting the distance to avoid the land (e.g. 'hug' the shoreline and attempt a surprise attack). There are many possible extensions to this research along these lines.
[3]

Another way we want to extend this research is through the implementation of a hierarchy of tactics. The tactics described in this research have been very low level - who to attack and where individual boats should move. In the future, we want to implement layers of the hierarchy, representing different planning levels. This would also involve multiple groups of boats similar to the group in our scenarios and approach the planning of strategy. We believe that equiping each boat in a group with an individual IM as described in this work, but adding a higher level IM for each group to be a promising approach to hierarchical planning. Instead of specifying what to attack and where to move, the group IM could provide tactics such as an area of the map that has higher importance for a particular group. By combining the group IM with the individual's IM, this could be implemented.

Overall we have found the research presented here to be an effective way of generating spatial tactics for groups of boats.[4]

## References


[1] C. Miles, "Co-evolving real-time strategy game players," Ph.D. dissertation, University of Nevada, Reno, 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] 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.


[2] rewrite: showed a high tendency to find rudimentary solutions

[3] rewrite: We would also like to investigate further representations of the leader as discussed previously.

[4] rewrite: The results have shown some very promising behaviour, and shows that this is definitely a promising research area. We look forward to improving and extending the research into a mechanism that is useful to a number of areas.

[5] ——, "Co-evolving influence map tree based strategy game players," in *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Games*. IEEE Press, 2007, p. Pages: to appear.
[6] ——, "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.
[7] 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.
[8] 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, p. Poster.
[9] C. Miles, "Lagoon craft," 2007.