Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Games (CIG 2007)

Co-Evolving Influence Map Tree Based Strategy Game Players

Chris Miles, Juan Quiroz, Ryan Leigh, Sushil J. Louis
Evolutionary Computing Systems Lab
Dept. of Computer Science and Engineering
University of Nevada, Reno
miles, quiroz, leigh, sushil@cse.unr.edu

***Abstract*— We investigate the use of genetic algorithms to evolve AI players for real-time strategy games. To overcome the knowledge acquisition bottleneck found in using traditional expert systems, scripts, or decision trees we evolve players through co-evolution. Our game players are implemented as resource allocation systems. Influence map trees are used to analyze the game-state and determine promising places to attack, defend, etc. These spatial objectives are chained to non-spatial objectives (train units, build buildings, gather resources) in a dependency graph. Players are encoded within the individuals of a genetic algorithm and co-evolved against each other, with results showing the production of strategies that are innovative, robust, and capable of defeating a suite of hand-coded opponents.**

Keywords: Co-Evolution, Game AI, Computer Game, Real-Time Strategy Games.

Fig. 1. TASpring

I. Introduction

While AI research has in the past been interested in games like checkers and chess, modern computer games are very different and have not received much attention from researchers [1], [2], [3], [4], [5]. These games are situated in a virtual world, involve both long-term and reactive planning, and provide an immersive, fun experience. At the same time, we can pose many training, planning, and scientific problems as games where player decisions determine the final solution.

Developers of computer players (game AI) for these games tend to utilize finite state machines, rule-based scripting systems, or other such knowledge intensive approaches. To develop truly competitive opponents these computer players often cheat, changing the nature of the game in their favor, in order to defeat their human opponents [6]. These approaches work well - at least until a human player learns their habits and weaknesses - but require significant player and developer resources to create and tune to play competently. Development of game AI therefore suffers from the knowledge acquisition bottleneck well known to AI researchers.

By using evolutionary techniques to create game players we aim to overcome these bottlenecks and produce superior players. Computer Real Time Strategy (RTS) games are of particular interest to us. These are games such as Starcraft, Dawn of War, TASpring (Figure 1), Company of Heroes, or Age of Empires [7], [8], [9], [10], [11]. In these games, players are given buildings, troops, and money. They play by allocating these resources: money is spent producing units and constructing buildings, and units are given various tasks to carry out. Units carry out these orders automatically, and the game is resolved by destroying other players' assets.

> "A good game is a series of interesting decisions. The decisions must be both frequent and meaningful." - Sid Meier

All games are fundamentally about making decisions and exercising skills. RTS games concentrate player involvement around making high level, long term strategic decisions. While varying greatly in content and style, RTS games are unified as a genre by a set of common foundational decisions. Most of these decisions can be categorized as either resource allocation problems: how much money to invest on improving my economy, what kind of troops to field, or what technological enhancements to research; or as spatial reasoning problems: which parts of the world should I try to control, how should I assault this defensive installation, or how do I outmaneuver my opponent in this battle. By developing systems capable of making these decisions, which are both challenging and relevant, we develop systems capable of tackling important real world problems.

RTS games have, by design, a non-linear search space of potential strategies, with players making interesting and complex decisions - many of which have difficult to predict consequences later in the game. We aim to explore this non-

linear search space of game-playing strategies by using genetic algorithms. Previous work has used genetic algorithms to make allocation decisions within RTS games, and has evolved influence map trees to make tactical spatial reasoning decisions within computer games [12], [13], [14]. In this paper we extend our influence map tree based system to play a complete RTS game, pulling back from the purely tactical level to look at more strategic decisions, while greatly complexifying the allocation decisions the player must make.

A. Game Player - IMAI Overview

Our game players, (which we call the influence map based artificial intelligence, or IMAI), play the game by casting it as a resource allocation problem. Solutions or allocations to this problem can be readily mapped into game-playing actions. Inside this generic architecture a variety of subsystems are at work, dealing with the many aspects of playing such a game. The spatial decision making system looks at the game world and determines promising locations to carry out various tasks - build a base here, attack your enemy's resources over there, cover your weak side over there. Spatial and non-spatial objectives are then chained into a dependency graph. For example, to capture points you must first train units, and to train units you must first build buildings. Expected benefit propagates from goal objectives to more immediate objectives, allowing the AI to judge the utility of these prerequisite objectives. Once resources have been identified and objectives have been defined, an allocation system does the bipartite mapping between the two: deciding that this group of units is in a good position to assault that enemy headquarters, while more money needs to be devoted to the construction of defenses around that bottleneck. Combined into a game player, these systems are capable of carrying out robust and coordinated strategies.

Designed to do more than just playing the game effectively, the IMAI uses generic systems for both spatial reasoning and allocation. This allows for effective evolution and co-evolution as each player can be encoded and decoded from a bit-string, the contents of which can lead that player to use a wide range of competent strategies. We represent possible game playing strategies within the individuals of a genetic algorithm's population. The game theoretic meaning for strategy is used here - a system which can choose an action in response to any situation [15]. We then play players against one another, using a fitness function which evaluates their in-game performance. This co-evolutionary search leads to increasingly competent players engaged in a constant game of one-upsmanship with one another in order to develop more robust strategies.

II. IMAI

IMAI players are capable of implementing a range of competent strategies within the context of RTS games. The IMAI works on the abstract level by casting the play of the game as a resource allocation problem. IMAI players run in a continuous loop while the game is being played as shown in Figure 2. Each iteration through the loop has 3 major phases:

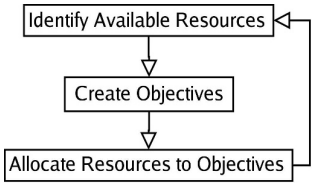

Fig. 2. IMAI Main Loop

resource identification, objective creation, and allocation. The processing is distributed, with a small amount of processing done each game tick. This keeps the game-player from being instantly responsive - an advantage not shared by its human opponent, and it helps avoid overburdening the CPU. In the resource identification phase the system analyzes the current game-state, determining which resources are available for the player to allocate. Most of this is trivial, such as fetching the amount of money the player has available, and listing the units and buildings not already occupied with some other task. In the objective creation phase several systems go to work, trying to determine possible goals for the player to work towards. The most complex category of objectives are spatial objectives - analyzing the world to determine where the player should attack, defend, build up, etc. The spatial reasoning system uses influence map trees (IMTrees) to determine these objectives - explained in detail in Section III. Non-spatial objectives include training units, constructing buildings, and determining the importance of gathering resources. Objectives are chained together in a dependency graph, propagating the expected benefit assigned to an objective to its prerequisites. If the AI wants to attack an enemy base it first has to build warships, to build warships it has to construct manufacturing facilities capable of producing them while expanding its economy to be able to afford them. Once the available resources have been determined and the desired objectives have been created, the allocator determines a good allocation of resources to objectives. This allocation can be trivially converted into commands used to play the game. In the next few sections we will discuss each section of the AI in more detail.

A. Identifying Resources

A resource is something that can be utilized to help achieve some objective; on the most abstract level the IMAI plays by allocating resources to objectives. The IMAI divides resources into four categories, three of which are easily identified: unit groups, which are collections of entities that can perform spatial tasks; builders, which are entities capable of constructing other entities (buildings and units); and generic resources, which are things like money, power, or wood which are necessary for realization of many objectives. The final category of resources considered are build points: areas on the map on which buildings can be constructed. To identify these locations we use a hand-coded simplification of the IMTree system used to do spatial reasoning in Section III.

B. Creating Objectives

An objective is a task which the player considers beneficial to accomplish. Each objective has two key methods, first it can determine the expected benefit for allocating any set of resources to it, second it can determine its own feasibility based on any set of resources allocated to it. We could reduce this to a single function with infeasible tasks returning zero benefit, but having two functions allows for more efficient allocation. The IMAI considers three categories of objectives: spatial objectives, which are points in the game world to attack, defend, or move units to; construction objectives, which are units or buildings the IMAI wants constructed; and resource gathering objectives, which are general priorities given to increasing the player's income. Each spatial objective is a (rawBenefit, task, location) tuple combined with a collection of enemy forces expected to provide resistance and meta-data specifying what types of units should be allocated [13]. RawBenefit is a value assigned to the spatial objective when it is created, specifying how much benefit is expected from accomplishing this objective. For spatial objectives, $benefit = rawBenefit * matching(units, metadata) * ratioOfStrength(units, resistance)$ and $feasibility = ratioOfStrength(units, resistance) > threshold$. The ratioOfStrength function takes two groups of units and based on the type, condition, armaments, and armor level present on each unit in both groups calculates a single real number representing the expected outcome. The creation of spatial objectives is done by the spatial reasoning system, and is detailed in Section III. Construction objectives contain the unit they would like to construct, and the benefit expected from constructing it. They are feasible if appropriate builder units or build points have been allocated, and if adequate money has been allocated. Each resource gathering objective contains the name of the resource it represents, and a single real number representing the priority associated with gathering that resource. Resource gathering objectives are used as placeholders in the objective chaining system; resources are not directly allocated to them.

C. Objective Chaining

Once the various objectives have been created, they are formed into a dependency graph, an example of which is in Figure 3. Only spatial objectives have self determined benefit. The benefit associated with training a unit is determined by calculating the expected benefit from having such a unit. For example, at the beginning of the game the spatial reasoning system usually produces many objectives related to capturing the neutral resource points near the player's town. Spatial objectives are chained to unit training objectives, passing on benefit proportional to how well those units match the objective. In Figure 3 the capture point objectives are passing a large amount of benefit onto the "train scout", as that is the best unit to accomplish this objective. In general, the chaining system propagates benefit from objectives to their prerequisites. The ultimate result is a collection of objectives, from places on the map to attack and defend, to construction and training orders. Each objective has an associated benefit, determined both from its own individual benefit and its relationship to the other objectives.

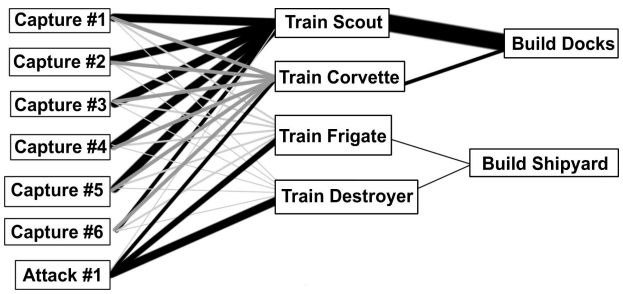

Fig. 3. Objective Chaining

D. Allocation Of Resources To Objectives

In previous work the allocation of resources to objectives was handled with a genetic algorithm [12], [13]. In this work we replace this system with a greedy allocator, primarily to reduce the computational burden required by frequently rerunning a genetic algorithm in the game. The allocator uses a simple greedy loop: find which resources are necessary to accomplish which objectives; take the resource, objective pair which has the highest $benefit/cost$ ratio; repeat. From there mapping the resource to objective allocation to in-game commands is trivial.

III. Spatial Reasoning

The most complex part of the IMAI, and the core of this research, the spatial reasoning system analyzes the game-state in order to produce a set of spatial objectives for the IMAI to carry out. The spatial reasoning system must provide a general representation of spatial reasoning, enough so to contain the wide variety of spatial strategies used by RTS game-players. To do this we use influence map trees (IMTrees), which are an extension to classical influence maps.

A. Influence Maps

An influence map (IM) is a grid placed over the world, with values assigned to each square by a problem specific function (IMFunction). Once calculated, each influence map relates some spatial feature or concept determined by the IMFunction. Influence maps evolved out of work done on spatial reasoning within the game of Go and have been used sporadically since then in various games such as Age of Empires [11], [16]. In an RTS game the IMFunction might be a summation of the natural resources present in that square, the distance to the closest enemy, or the number of friendly units in the vicinity. The motivation behind influence maps is that each IM is easy to compute and understand, and that they combine together in intuitive ways to perform complex spatial reasoning. Figure 4 is a visualization of an influence map, with the IMFunction being the number of triangles within some radius. If each triangle was a resource location, and the radius of the circle

was the effective resource gathering distance, this IM could be used to find optimal resource gathering locations for any situation. Traditionally influence maps were carefully hand-coded and used to solve particular problems. A small set of IMs are created and then combined in a weighted sum to produce the desired IM. In our resource gathering example, we could add an IM where the IMFunction = inverse distance to friendly buildings, on top of the existing nearResource IM. Summing them together produces an IM containing good resource gathering locations near existing structures, leading to effective player expansion.

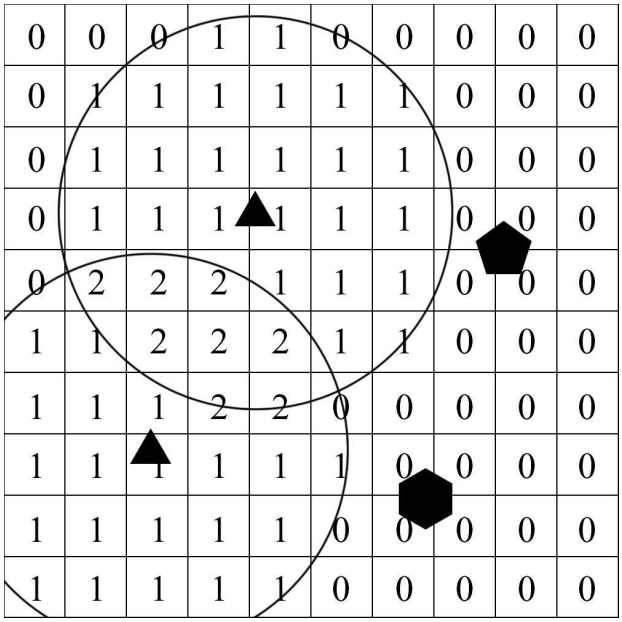

Fig. 4. An Influence Map

B. Influence Map Trees

We contain IMs within a tree structure instead of the traditional weighted list [16]. The goal being to have a more general structure that can be effectively evolved. Each tree represents a complete decision making strategy, and can be encoded as part of an individual in a genetic algorithm. Leaf nodes in the tree are regular IMs, using basic functions to generate their values based on the game-state. Branch nodes perform operations on their children's values in order to create their own values. These operations include simple arithmetic operators: combining their children's values in a weighted sum or multiplication to form new values. These nodes can also perform processing on the values of a single child, smoothing or normalizing their values. Many game AI developers use specialized post-processing methods to manipulate and customize their influence maps. For example, Age of Empires uses multi-pass smoothing on influence maps to determine locations on which to construct buildings - almost identical to how our build locations are determined. By allowing nodes in our tree to perform such processing methods, a single IMTree can concisely represent the variety of influence map based game-playing strategies hand-coded within many other game AI systems.

Each IMAI possesses several IMTrees, with each tree representing some category of spatial reasoning - places to attack, or places to defend. To create objectives from the game-state the IMAI does a post-order walk on its IMTrees, letting each node calculate its values. Then, an objective zoner analyzes the root IMs, producing a list of spatial objectives. It creates a spatial objective at each local optima in the IM, with rawBenefit equal to the value of the IM; this is explained in more detail in [12], [13]. By encoding and evoling IMTrees we create a technique similar to that of GP, but in a spatial domain.

IV. The Game - Lagoon

We developed Lagoon, a Real-Time 3D naval combat simulation game, as a platform for this research. Figure 5 is a screen-shot from a game between two evolved IMAI players. In this example, the blue player, who is coming from the top right, has pushed a heavy destroyer through his opponent's line and has flanked around to the left side of the screen. His opponent, the red player, is trying to hold that line with smaller boats while his capital ships pull back to engage the blue destroyer. Lagoon follows standard RTS paradigms for most of its game-play. It differs in its naval setting and in its relatively complex physics model. Like most RTS games, Lagoon uses a hierarchical AI system to distribute the work. At the top level each side has an IMAI, which makes broad sweeping orders that are passed down to the lower level AIs. In the middle level squad level AI managers coordinate groups of boats while subdividing major tasks (attack-move across the map) into smaller more manageable ones. At the lowest level behaviors carry out immediate tasks, maneuvering around boats to avoid fire, avoiding land, and staying in formation; all within the complexities and constraints of the physics model.

Lagoon has game-play similar to most other RTS games: players gather resources, train units, build buildings, and then send out their units to conquer the world. Lagoon has two types of resources: oil, which is gathered by capturing resource points; and power, which is generated by constructing power generators. Resource points are captured by stationing units within their vicinity for a few seconds. Captured points continuously produce income for their owner, allowing them to construct more units and buildings. The competition to capture and defend points is the driving factor behind the game-play, forcing players to compromise between expanding out and building up.

Lagoon has a variety of units, from small, quick assault boats to frigates, cruisers, and destroyers. Players must carefully balance the types of units they construct - taking advantage of their opponent's weaknesses while covering their own. Much of the strategy in playing an RTS game comes in finding the proper balance between the many simultaneous tasks being carried out: attacking enemy units and buildings, capturing and defending resource points, and building up to get more powerful units.

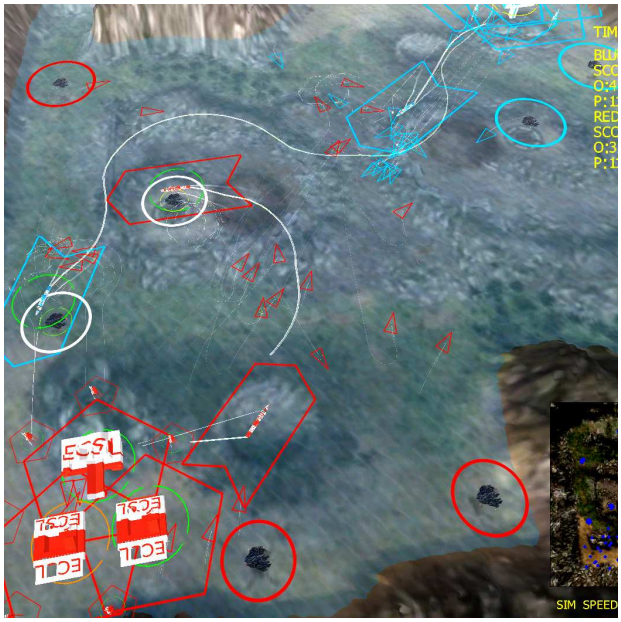

Fig. 5. Lagoon

A. The Mission

For this research we are evolving players to play 1 on 1 games on a single map. Figure 6 is an overhead shot of the map we are playing. The map is symmetric, with players starting in opposing corners, and resource points overlaid with white circles.

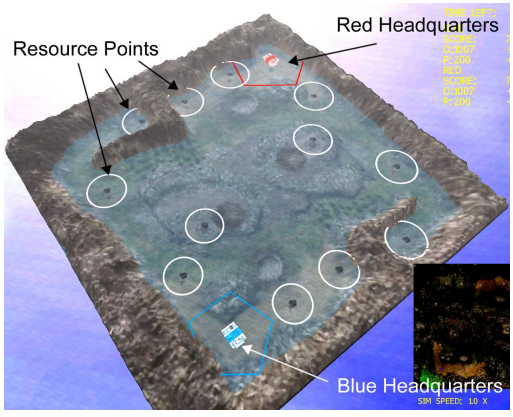

Fig. 6. Mission

V. Co-Evolution

Instead of hand-coding our AIs we use co-evolution to evolve them. This allows the production of truly innovative strategies, eliminates the need for expert knowledge, and leads to ultimately superior players. To carry out this co-evolution we are using a parallelized, queue based, steady state genetic algorithm, the details of which are beyond the scope of this paper. In short, we maintain a population of IMAI players, continuously playing them against one another to determine which ones are superior. Occasionally we replace poorly performing players with offspring from better performing players using standard genetic operators of roulette-wheel selection, one point crossover, and bitwise mutation. Crossover takes place with 75% probability, and the bitwise mutation probability was chosen to give on average 2 bit mutations per child. Individual matches between two IMAI players are resolved by playing the full game, and fitness is determined by playing the game and analyzing the results.

A. Encoding

The GA packs all the parameters for each IM in the IMTree into a bit-string, with fixed point binary integer encoding for the enumerations and fixed point binary fraction encoding for the real valued parameters and coefficients. The GA does not directly have control over the rest of the IMAI system, but we have found that by tweaking various aspects of the spatial reasoning system it can achieve an amazing variety of effects - biasing it to build up the tech tree to powerful units immediately, or just amassing a horde of the cheapest boats and then flooding their opponent. At this phase we were not evolving the structure of the tree, purely the parameters and coefficients for each IM. The influence maps use the same basic structure as our hand-coded players, however a very wide range of behavior has been observed in the players strategies.

B. Evaluation and Fitness

To evaluate each individual we play them against an opponent and examine the results of the match. Whichever player gathers the most resources within ten minutes is considered the winner. Note: this is not the amount of resources saved up, but the total amount of income that was gathered without regard for how it was used. It was empirically noted that the ultimate winner was usually the player that gathers more resources, and that the ultimate result could be estimated fairly accurately after ten minutes. We use fitness sharing to adjust each player's fitness, so that the amount of fitness a player gains by defeating an opponent is inversely proportional to how many other players have defeated that opponent. By defeating a player no one else can beat, a player can have a high fitness, even if it looses all other games. Fitness sharing encourages and protects diversity within members of the population.

C. Hand-Coded Opponents

We develop three hand-coded opponents against which to test our AIs. Each plays a different, but effective strategy. The first player plays a solid rushing strategy, constructing a basic manufacturing building and then training large numbers of the basic combat ship. It attacks in force early in the game, trying to destroy the enemy's base while the enemy is unprepared. The second player plays a solid defensive strategy, expanding out quickly early in the game, and then concentrating most of its units on defending its points. It quickly builds up to powerful units, attacking when an overwhelming force is available. The third player plays a more balanced game, capturing

points continuously while trying to assault the opponent's base and points. It will aggressively pursue its advantage if it gets momentum, pushing back its opponent's lines until it reaches their town. In testing between the three hand-coded players, the balanced player was the most effective, as the two other hand-coded players' strategies are less flexible. If the rusher fails in its initial rush, its economy is left in a weakened state, making the player vulnerable to enemy attacks. The defender on the other hand does not expand aggressively in the late game, which helps preserve its units, but generally costs it the win at the 10 minute point. Matches between aggressor and defender usually resulted in stalemates, with both sides massing large armies they are unwilling to commit to action.

VI. Results

We create a population of 25 random individuals, which are tested against each other and evolved as described in Section V. After every 25 evaluations, 5 players are sampled at random from the population to play against each of the hand-coded opponents. The IMAI players are not rewarded or punished for winning or losing against the hand-coded players, it is purely used as a benchmark to see how the population is evolving over time. We graph the total score our evolved players receive against the static AIs in Figure 7.

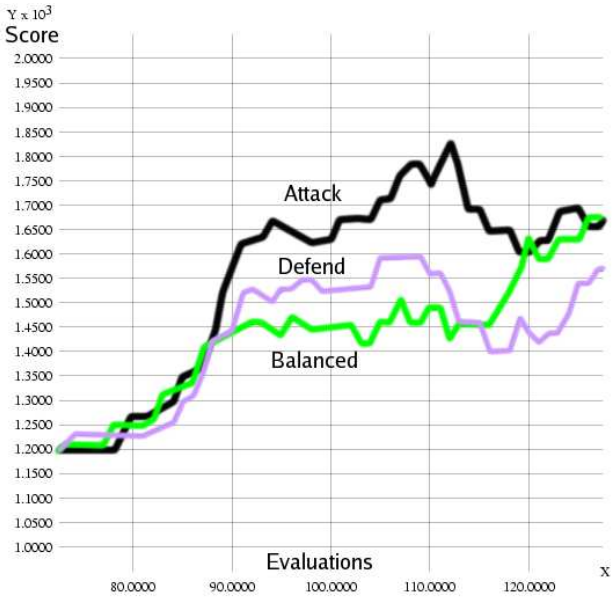

Fig. 7. Scores received by Co-Evolved players against Hand-Coded AIs

We see a solid improvement in scores throughout the course of co-evolution - evidence that our co-evolutionary arms race is producing increasingly better players. The players score the best against the aggressive opponent, which is reasonable because the attacker generally sacrifices expansion and capturing points in order to attack its enemy's town. So long as the IMAI players keep their town well defended, they should be able to capture most of the map and score unnaturally high.

The score is an approximation for how well they are playing, whether they are actually winning those games is another matter. Taking a win as a score of 1.0 and a loss as a score of 0.0, we calculate averages and graph as before in Figure 8.

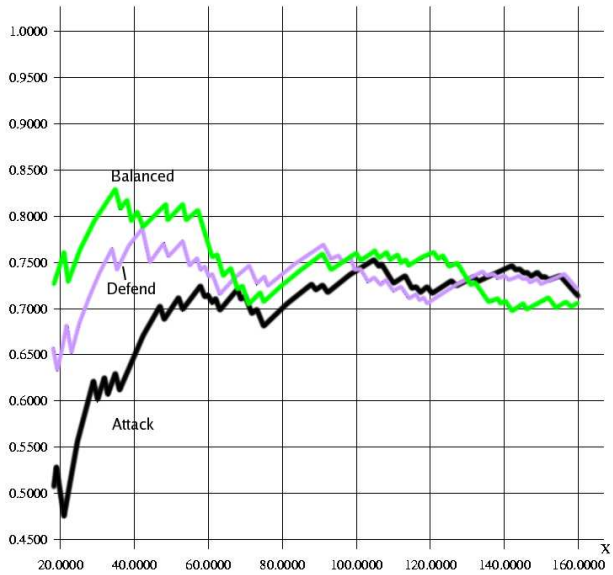

Fig. 8. Winning Scores against Static Opponents

The IMAI players very quickly became superior to the hand-coded players, within the first few generations of evolution. Analysis shows that most IMAI players play very similarly to the balanced player, only with superior coefficients controlling how they balanced their allocations. They generally attack their opponent's economy as well, as opposed to the balanced AI which tends to target manufacturing facilities. An example is shown in Figure 6 where an evolved player is flanking its opponent's line to push in and disrupt its points. We created a large set of random strategies and then saw how they played, noting that the large majority of non-lethal players (at least somewhat competent) used strategies similar to the balanced AI. Even in a few generations of evolutions players have advanced to the point where they are better then our best hand-coded players. Against the defensive player the IMAI has some initial issues, because it is a strategy under-represented in a random population. The defensive strategy is the weakest of the three and as the IMAI players develop more robust strategies they start to soundly beat it. Against the attacking strategy the IMAI has initial problems, as with the defensive strategy all out rushes are under-represented in a random population. Unlike the defensive strategy however, rushes are very effective against unprepared opponents. While the attacking strategy was beaten by both of the other hand-coded strategies, it was initially the most effective against the IMAI players. Later in co-evolution the players become exposed to evolved rush-like strategies, and they develop counter strategies for beating them. We see continuous improvement in the scores players receive, with their probability of victory increasing

more slightly. This is across all three types of opponents, showing that the evolved IMAI players are robust enough to work against a variety of opponents.

Under further analysis we discovered why the scores continue to improve but the overall win rate is relatively stagnant. The most obvious and remediable problem is that we are randomly sampling individuals in the population to test against static opponents, by taking individuals with above average fitness we could likely get significantly higher performance. Virtually all games played go the ten minutes until one player has gathered more resources. It is just too short a period of time to overwhelm your opponent and destroy their base. We found that the IMAI evolved players would concentrate their spending very heavily on improving their economy, building an excess of power generators, and occasionally headquarters in order to improve their resource income. They would then slowly lose resource points to their opponent, who had allocated money to training troops, in order to buy time until the ten minutes was up. Then, even though they were really losing the game they'd be declared the winner.

We also noted that while over the long term the win/loss ratio was a slow increase, over shorter periods of time the population cycled through a variety of strategies. We see these cycling behaviors, players that use small boats are better against the attacker and defender, while players that use large boats are better against our balanced player. Graphing wins averaged over shorter periods of time shows this cycling effect - Figure 9. IMAI players tend to cycle through the types of units they prefer during evolution. In the beginning most players use small units, and players that use larger boats are more effective and start to dominate the population. This continues until most players concentrate hard on the largest boats. However, the largest boats can be overwhelmed with a large number of the smaller boats, so the cycle eventually circles back around to smaller boats. Fitness sharing protects the various species, but the population size is too small and the number of evaluations appears to be too low to reach equilibrium. Since the hand-coded players are static, during times when the IMAI players use the appropriate counter units they win a large percentage of the games. Conversely, during times when they use the wrong units they lose a large percentage of the games. Future testing will increase the population size, which should remedy this problem.

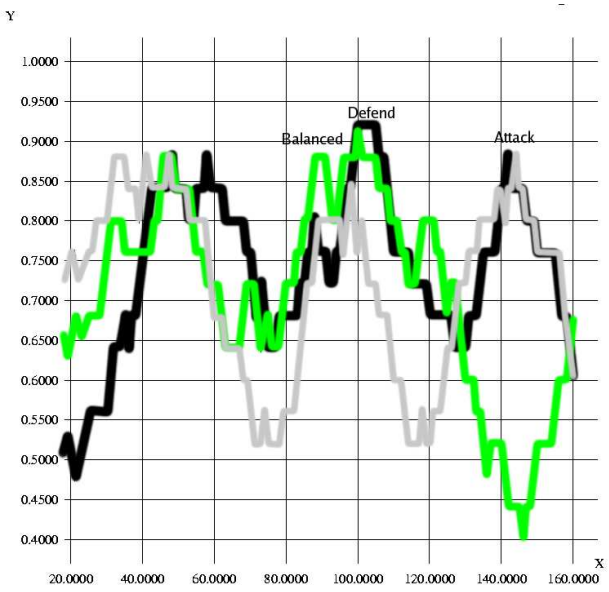

Fig. 9. Winning Scores against Static Opponents - Less Averaging

VII. Conclusions and Future Work

Co-evolution produced IMAI players who where significantly superior to our hand-coded players. These players played strong, robust strategies that were effective against a variety of opponents. Most co-evolved players used strategies similar to that of the balanced AI, simultaneously attacking and defending across a contiguous front. Unlike the balanced player however, the IMAI players were capable of attacking various points on the front in order to take advantage of opponent weaknesses, gradually cornering the opponent and then pushing in with heavy units. Against humans the IMAI was very effective, soundly beating everyone we could talk into playing it. Future work would include testing just how effective they are against human opponents, and test a number of human opponents against a variety of opponents from various stages in the evolution.

The IMAI architecture is highly general. By virtue of design, new game-playing functionality can be efficiently added and modified. The system should be adaptable to other RTS games, as well as to a range of real world problems.

In previous research the IMTree spatial reasoning system was shown to be highly general as well, with strategies learned on one map being effective across a wide range of situations. Future work would show that this continues to be true, by testing players evolved on a single map against those evolved on multiple maps.

There still a few common aspects of RTS games our IMAI does not know how to handle. One is research, which players perform in order to enable units or upgrade their abilities. This should be an easy addition to the objective chaining system. Second is walls, which will be an odd fit in the naval setting but are an important part of many RTS games. Third is special unit abilities or heroes, where units can occasionally perform powerful abilities. This is generally more of a tactical than a strategic decision, but it could have an impact. Implementing all of these within our game, and extending the IMAI to deal with them would show generality to virtually all RTS games. We also disabled the IMAI players from determining their own building locations as they would frequently construct buildings in their opponent's town - which while a legitimate strategy in many RTS games, it became an overpowering strategy in Lagoon.

The major avenue for future work is to do comparative studies between the effectiveness of our AI and other AIs.

Some games such as TASpring or Empire Earth allow for the integration of new AI systems. It would be very interesting to see how evolved IMAI players stack up against industry quality AI opponents.

Fig. 10. Supreme Commander

VIII. Acknowledgments

This material is based upon work supported by the Office of Naval Research under contract number N00014-05-0709.

References

[1] P. J. Angeline and J. B. Pollack, "Competitive environments evolve better solutions for complex tasks," in *Proceedings of the 5th International Conference on Genetic Algorithms (GA-93)*, 1993, pp. 264–270. [Online]. Available: citeseer.ist.psu.edu/angeline93competitive.html

[2] D. B. Fogel, *Blondie24: Playing at the Edge of AI*. Morgan Kauffman, 2001.

[3] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, pp. 210–229, 1959.

[4] J. B. Pollack, A. D. Blair, and M. Land, "Coevolution of a backgammon player," in *Artificial Life V: Proc. of the Fifth Int. Workshop on the Synthesis and Simulation of Living Systems*, C. G. Langton and K. Shimohara, Eds. Cambridge, MA: The MIT Press, 1997, pp. 92–98.

[5] G. Tesauro, "Temporal difference learning and td-gammon," *Communications of the ACM*, vol. 38, no. 3, 1995.

[6] J. E. Laird and M. van Lent, "Human-level ai's killer application: Interactive computer games," 2000. [Online]. Available: http://ai.eecs.umich.edu/people/laird/papers/AAAI-00.pdf

[7] Blizzard, "Starcraft," 1998, www.blizzard.com/starcraft. [Online]. Available: www.blizzard.com/starcraft

[8] R. E. Inc., "Dawn of war," 2005, http://www.dawnofwargame.com.

[9] "Taspring," 2006, http://taspring.clan-sy.com/.

[10] R. E. Inc., "Company of heroes," 2006, http://www.companyofheroesgame.com/.

[11] E. Studios, "Age of empires 3," 2005, www.ageofempires3.com. [Online]. Available: www.ageofempires3.com

[12] 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, p. Pages: to appear.

[13] ——, "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, p. Pages: to appear.

[14] S. J. Louis, C. Miles, N. Cole, and J. McDonnell, "Learning to play like a human: Case injected genetic algorithms for strategic computer gaming," in *Proceedings of the second Workshop on Military and Security Applications of Evolutionary Computation*, 2005, pp. 6–12.

[15] R. Gibbons, *Game Theory for Applied Economists*. Princeton University Press, 1992.

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