Adapting to Human Game Play

Phillipa Avery, Zbigniew Michalewicz

***Abstract*— No matter how good a computer player is, given enough time human players may learn to adapt to the strategy used, and routinely defeat the computer player. A challenging task is to mimic this human ability to adapt, and create a computer player that can adapt to its opposition's strategy. By having an adaptive strategy for a computer player, the challenge it provides is ongoing. Additionally, a computer player that adapts specifically to an individual human provides a more personal and tailored game play experience. To address this need we have investigated the creation of such a computer player. By creating a computer player that changes its strategy with influence from the human strategy, we have shown that the holy grail of gaming – an individually tailored gaming experience, is indeed possible. We designed the computer player for the game of TEMPO, a zero sum military planning game. The player was created through a process that reverse engineers the human strategy and uses it to coevolve the computer player.**

I. Introduction

A common method of representing a computer player is by a static strategy for game play. The representation of the strategy could be a neural network, a set of *IF – THEN* rules (fuzzy or not), decision trees or many other means. Regardless of the representation, the use of a static strategy results in a computer player that becomes obsolete once the human player adapts to it. When a computer player ceases to challenge the human player, it is no longer fun to play against. Thus, a game play experience that is unique depending on the circumstance and the human game strategy used has been of significant importance in recent years. Games such as Fable (Lionhead Studios) and Star Wars: Knights of the Old Republic (BioWare), which change the game scenario dependant on the choices the player makes (to be a good or evil character), have been very successful. The 'choose your own adventure' style has proven to be a lucrative venture, and adaptive adversaries are key to continued success. It seems that the ability for a computer player to adapt to an individual human player is the holy grail of game play.

The adaptation to the human has a number of advantages. Firstly, it grows in strength with the human. When the human player starts playing the game, they are weak in strategy creation for the game. They then start to increase in strength as their understanding of the tactics increases, and they can develop strategies that are more complex. The same can be said for the coevolutionary algorithm. The initial generations of a coevolutionary computer player create very basic and not overly intelligent rule development. Then, as each game is played, it has a chance to encompass and counter the oppositions strategy of game play and create better rules. We wanted to use this potential of the coevolutionary system to create a computer player that coevolves to adapt to a human player relative to the humans capability. By including the human strategy into the coevolutionary system, the system can evolve strategies that have adapted to the human strategies. As the human gains more experience in playing the game, so does the computer player.

Additionally, the ability to provide an adaptive computer player that tailors itself to the human player has a lot of potential in the training industry. It is now fairly well accepted that playing computer games is beneficial for teaching purposes [1]. The ability to provide a fun and challenging way of learning has clear benefits. The problem lies in finding a way to provide the individualistic training for each student. Currently this lies in providing a level rating (e.g. easy, normal, and hard) that the player can choose. This has difficulties however, as the classification of student levels into (normally) three levels of expertise has obvious disadvantages. Instead, by providing an experience that is tailored at a specific human player, the player can then experience a game that simultaneously gains in difficulty as their experience and skill in the game increases.

To achieve the goal of creating an adaptive computer player, we needed to create a way of including the human strategy in a form the coevolutionary system could understand and use. This essentially involved creating a method of reverse engineering the human strategy from the outcome of the game-play. We decided to create an iterative system that would record the data (the human choices and the game environment) from each game played, and use it to create a model of the human. The model would then be in a form that could be included in the coevolutionary process. The human model is then added as a supplementary individual to the coevolutionary populations, so that the coevolution can evolve with, and against the model.

By reverse engineering the human's strategy into a set of rules and adding the model created to the coevolutionary system, we are able to influence the coevolutionary process. At the beginning of the human's learning progress, he or she will not have any experience in playing the game, and are likely to have minimal strategy development. This stage is probably the most difficult to reverse engineer, as the human is more random in strategic choices. Consider the idea of beginner's luck in games such as poker; expert players can have trouble reading beginners, as the beginner makes seemingly random choices due to inexperience. However, even the very general (and probably not effective) rules

P. Avery is with Dept. of Computer Science, University of Adelaide. Z. Michalewicz is with the Dept. of Computer Science, University of Adelaide; also at the Institute of Computer Science, Polish Academy of Sciences, and Polish-Japanese Institute of Information Technology (email: pippa, zbyszek@cs.adelaide.edu.au).

978-1-4244-2974-5/08/$25.00 ©2008 IEEE

reverse engineered at this stage have a chance of affecting the coevolution. It is unlikely that the human rules will be considered the best individual in the population for elitism, but they still have a chance of effecting the next generation through selection and crossover.

As the human starts to gain experience in the game, he or she develops 'winning' strategies. It is likely that the human will then repeat the same or similar strategies over concurrent games if the strategy continues to work. It is here that the adaptive coevolutionary system really comes into play. As the humans form a clearer pattern for reverse engineering, the rules being modelled and added to the coevolutionary system have a greater impact. Now the coevolution can act to directly overcome the human strategy, creating new strategies that are of greater challenge to the human player. This creaates a system that grows and improves along with the human the computer player is competing against; a tailored system that provides the best challenge for the individual human.

This paper discusses the issues involved with creating a computer player using coevolution that can adapt to humans, as well as the methods we used to create this system. We begin with a brief description of the game of TEMPO, followed by a review on the current research in adapting to humans. We then discuss the mechanism used to create our adaptive computer player. After implementing this system, we ran a user study to observe effectiveness, as well as the way human players interact with the computer player. The results of this user study are provided, along with discussion and analysis. We conclude the paper by discussing the findings, and the areas of research that have been identified for future work.

II. The TEMPO game

Tempo is a zero sum game played between two opposing parties by allocating resources in a cold war style simulation. The goal of the game is to acquire more offensive utilities (utils) than the opposition before war breaks out. The decision making process requires allocating a yearly budget resource between offensive and defensive weapons of various strength and cost. The purchase of intelligence is also provided to give insight into opponent's allocations, and counter intelligence is provided to negate this intelligence (to a degree). For more information on how the intelligence and counterintelligence aspects work, see [2].

As a brief summary, the game involves yearly iteration of budget allocation until a randomly chosen number depicts that war breaks out and the game ends. The chance of war breaking out is given as a constrained randomly increasing probability for each year. The budget also increases at a constrained random amount for each year. The budget must be allocated to the purchase of intelligence/counter intelligence, and/or the defensive and offensive weapons. Any budget not spent in the year is lost.

Each weapon is either offensive or defensive, with offensive weapons countering defensive and vice versa. Further subtypes exist of A and B. Offensive A weapons counter Defensive A weapons, but not Defensive B. Each weapon is then given a unique ID, such as Offensive A1 (OA1). The weapons have their own unique parameters, such as the cost to buy a single unit, the cost to operate (maintain and use) a unit, the power (utils) and so on. At the end of the year, all weapons that have been operated for a given category/type are summed and the opposition counter category subtracted. This gives the net offensive and defensive utils for the player. When war breaks out, the player with the largest net offensive utils (OA + OB) wins the game.

III. Research on Adapting to Humans

While there has been much research on evolving computer players to beat human players, more is needed on computer players that are designed to be *challenging* for humans. This means creating computer players that are specifically tailored to give the human player a challenge, not just trying to find the optimal way to beat them or lose to them every time [3], [4]. To find a way to adapt to a human player, we investigated two areas; ways to extract rules representing a human's strategy from the output of their game play, and ways the system can use these rules to adapt.

The most relevant research on this topic is being performed by Louis et. al. with their Case-Injected Genetic Algorithm (CIGAR) research [5], [6], [7], [8], [9]. The CIGAR system uses a database (the case base) of problems mapped to solutions (cases) to prompt the Genetic Algorithm (GA) to come up with better and more human-like solutions. The research to apply CIGAR to games was performed by using a strike force Real Time Strategy (RTS) game, where the computer player has to allocate their resources to a set of aircraft platforms that attack the human players forces (offensive and defensive buildings for aircraft warfare). The CIGAR system for this includes cases learnt from humans in previous games, storing the human moves in the case base. Thus, the case base consists of human derived cases as well as the cases discovered by the GA from playing against human and computer players. The cases are then chosen using a similarity metric, and there is a possibility of a human case being chosen from the case base.

This approach differs from ours in a number of ways. Firstly, our system uses a coevolutionary mechanism, not a GA. This changes the way it can use the memory, and adapt to human players. The coevolutionary system can be continuously run without any human interaction, and is constantly changing and adapting. There is no search for an overall optimum, just a way to beat the current opposition. There are also differences in the way individuals are represented. Our TEMPO coevolutionary system uses individuals representing a strategy in fuzzy logic. To use human knowledge in the TEMPO system, the human strategy must also be represented as a fuzzy rule base, which can be difficult to do. This relates to another difference in this research, that of the differences in the game itself. The TEMPO game is not a RTS game, it is a turn-based game where each player makes his or her decisions simultaneously. Each year of game play, the environment of the game changes and becomes more complex, and the knowledge on what

the opposition is doing is minimal and can be misleading due to the use of intelligence and counter intelligence. The players being developed in the TEMPO system are developed to encompass all this and develop generalised strategies for game play. Additionally, the adaptive system we are proposing is designed to adapt to a single player, creating a real-time tailored experience for that player. This goal of a tailored learning experience differs from the current CIGAR research conducted. Our goal requires that the human rules have to be obtained from game play with, and added to, the currently evolving system.

Other relevant research by Ponsen et al [10], [11], [12] extends the dynamic scripting method developed by Pieter Spronck et al [13], [14], [15] for RTS games. The dynamic scripting is used to change the strategy rules (the script) for an opponent during game play. Rules that perform well in a particular dynamic situation are given a higher weighting, with greater chance of selection. The rules themselves are manually designed for the specific game being implemented (similar to the way most current AI for games is done). Ponsen extends the dynamic scripting to RTS by changing the script during successive stages of the game as more resources become available. In addition, an off-line evolutionary algorithm was applied that attempted to create scripts to counter well-known optimized tactics (for the game of WARGUS). Static players were used to measure against and the fitness was adjusted for losses and wins against the static player.

The disadvantages of using a static player to develop the scripts were addressed to some degree by Aha et al. [16]. Aha used case-base reasoning to select scripts for game play against random opponents chosen from 8 different opponent scripts. The evolutionary algorithm developed by Ponsen et al. was used to develop scripts for each of the eight opponent scripts, and the case-based system then chose which tactic to use (from possibly different scripts) at each stage of the game. The research using dynamic scripting does not use coevolution, and the mechanisms for the research are very different to ours. The idea of a set of static scripts is also very different to the way our adaptive system works, where our 'scripts' are developed on the fly.

Lastly, the idea of coevolving against human players was investigated by Funes et. al. [17], [18]. In this work, the game of Tron was used to evolve against human players on the internet. The game-play involved the human playing against a strategy developed using Genetic Programming (GP) coding. The evolution process contains two separately evolving populations. The foreground population contains a population of 100 individuals that play against human players for evaluation. Computer players are randomly selected from the population for game-play. Each generation the worst 10 individuals are replaced by the top 10 from the background population. The individuals are evaluated by playing a set number of games (5 for the 90 'veteran' players, and 10 for the 10 'new' players) against human players, the results of all games played against a human are then used to calculate the fitness. The second population (the background population) is used to evolve strategies through self play, training a population of 1000 individuals against a training set consisting of the best individual from the foreground population, and the top 24 individuals from the previous background population generation.

The research using Tron showed how coevolution with humans can be effectively used. However, the learning of the foreground population was very slow (with at least 550 games needed each generation). We are using coevolution to adapt real-time (after every game played) to a single human player, using a method of human opponent modeling to coevolve with. The method of our coevolution is also different, without a direct evolution against the human player, and with two populations of individuals performing self-play to learn strategies. The previously discussed differences in game and individual representation also apply. Also, our testing involved a controlled test environment, which online learning can not readily achieve.

IV. Coevolving with humans

Playing a statically coevolved player against the same human repeatedly allows the human to determine a counter-strategy that is dominant over the static computer player. The first time the human player plays against the strategy however, it is unknown to him/her and could possibly be difficult to beat. The question then arises as to whether the need for adaptation is indeed necessary. Could we not just continue the coevolutionary process and pick different individuals to play against the human each time? There could well be enough difference in strategy represented through the coevolutionary process itself to provide a challenge for each new game played.

While there has not been much recorded research on this topic, intuitively it would seem that a similar occurrence to the static scenario would eventuate. The reasoning behind this is that through our research we have observed the TEMPO coevolution reaching a plateau where the baseline measurement technique used displays results that show very little change (for more information see [19], [2]. Even though it is constantly changing and undulating, the evolution does not tend to make great leaps in development. This means that even though the human player would be playing a different player each time, the strategies being developed by the player are similar in strength to previous ones, and the human player could learn to overcome them. Additionally, the tailored system allows the players to directly counter the human strategy making, and hence give a much greater individual challenge.

There are a number of ways that the human models could be used in the coevolutionary process. Originally we thought of having a separate human population, similar to the memory population in previous research. However, the extra processing time required for selecting and evaluating a separate population was deemed excessive for the purpose. Additionally, if evaluation were the only influence the human model had on the populations, poorer human strategies would have little to no effect when they are constantly beaten.

Including the human models into the populations has a direct effect on the individuals being created (through selection and crossover). To include the human in the process, we needed to find a way to coevolve against the human model and the other randomly evolving players. This allows the system to create players that are still finding randomly evolving strategies that can take into account, and counter, the human player's actions. By including the human model in the coevolutionary process, when a new 'best' individual is chosen from the system to play against a human player, this individual has been able to incorporate and counter the stronger elements of the human strategies.

V. Representing human strategies

Representing (modelling) humans is a research field in itself and can be very difficult to do. To minimize this issue, we chose to very loosely reverse engineer the human choices as a model of the strategies used. The model used would also need to be of the same format as the coevolutionary individuals in the TEMPO system, which uses fuzzy logic rulebases. Doing this allows the human model to be directly inserted into the coevolutionary system for the process described above.

The reverse engineering of the human works as follows. When a human plays a game against the computer player, the data of the game is recorded. This data includes the choices the human made, and the environmental data for each game year. The data is then used in an evolutionary system to find individuals that model the human by mimicking the human choices. To evolve the human model, individuals represented in the same way as the coevolved individuals are randomly initialized. Each individual then plays the exact same game as the human, against the same computer player as the human played. The individual is evaluated as the difference in outcome and allocated resources to what the human achieved. The closer the individual comes to the same results as the human, the better the fitness.

We implemented the human reverse engineering mechanism using an evolutionary algorithm with a single population. The variation operators used were two point crossover and mutation where chosen genes were replaced with a random value. To avoid premature convergence on a sub-optimal solution we also forced the individuals to have unique genotype. We experimented with selection operators, and used ranked selection with elitism in the final process. After much experimentation and manual changing of the parameters to determine a good result, the final evolutionary parameters chosen were as follows. The process ran for 150 generations, with a population of 100 individuals. An elitism ratio of 5% was used, with a 50% crossover ratio, and a 10% mutation ratio. No rule penalty was applied to minimize the rules used in the rule bases, as this seemed to occur naturally.

The evaluation function was created by finding the difference in actions and results to the human player. Additionally we added changeable constant weightings. The weightings were applied to the differences in the outcome, weapons bought and intelligence/counter intelligence bought between the individuals and the human value. By adding weights, we can sway the evaluation importance of each of the parameters to create rules that are more realistic. For example, it might be that getting a closer outcome (total offensive utils at the end of the game) to the human was more important than creating rules that allocated the resources in the same manner as the human, and vice versa. Hence the evaluation function has the following evolutionary variables:

1) *human_NetUtils* – the total net offensive utils for the human player at the end of their game.
2) *individual$_i$_NetUtils* – total net offensive utils scored (playing the same game).
3) *Years* – total number of years played before war broke out.
4) *human_IntelChoice, human_OptChoice, human_BgtChoice* – data arrays of human allocations made for the year.
5) *individual$_i$_IntelChoice*, *individual$_i$_OptChoice*, *individual$_i$_BgtChoice* – allocations the individual made for the year corresponding to the human allocations.
6) *IntelWeight*, *WeapOptWeight* and *WeapBgtWeight* – constant weights applied.

The described evaluation function *eval* is implemented as

$$\begin{aligned} eval(individual_i) = {} & abs(human\_NetUtils - individual_i\_NetUtils) \\ & NetUtilsWeight) + \\ \sum_{t=1}^{Years} (& (abs(human\_IntelChoice_t - individual_i\_intelChoice_t) \\ & IntelWeight) + \\ & (abs(human\_OptChoice_t - individual_i\_OptChoice_t) \\ & WeapOptWeight) + \\ & (abs(human\_BgtChoice_t - individual_i\_boughtChoice_t) \\ & WeapBgtWeight)), \end{aligned}$$

We experimented with different values for the constants. For the final process, the *outcome* constant was assigned the highest preference with a weight of 5, followed by the *weapon* constant with a weight of 3. The *intelligence/counter intelligence* constant was given a weight of 1.

The rules evolved using this method give a rough estimation of possible strategies the human used. It by no means represents the human strategy exactly, which is in many ways a good thing. Our task is not to try and create an optimized computer player against a human player, but to create a computer player that is *challenging* for the human player. Even if it is only evolving against a rough estimate of the human player, for a single game-play situation, the evolutionary process is still given the opportunity to counter the human strategies.

VI. The human adaptive coevolutionary process

To incorporate all the ideas described above, we needed to develop an entire system for game play. We named this system the Human Adaptive Coevolutionary Process (HACP). HACP uses a coevolutionary algorithm comprised of two populations of individuals, and a memory population used to evaluate against. The individuals in the coevolution consisted of fuzzy logic rule bases, representing a particular strategy for game play. The evolution used two point crossover and mutation. Crossover was applied at rate of 30%, and if crossover was not applied to the individual, mutation was applied for each gene at a rate of 30% with a 10% chance of a large mutation. The generation number and population size are discussed later. The evaluation function was the won ratio (over all games played) for each individual. It was calculated by playing against a random sample set from

the opposing population for 5 games, then an additional 10 games against the memory population. An additional penalty was also applied to restrict the number of rules used. For more information refer to [19].

The coevolutionary system described was then given a simple graphical user interface to play against human players. This was then combined with the human reverse engineering (modelling) system. The process consists of the following steps:

1) The process begins by coevolving two populations against each other and the memory population.
2) After a set number of generations, the best individual from the currently winning population is chosen and played against the human.
3) The data from this game is then recorded and passed to the evolutionary human modelling system.
4) The modelling system then evolves a rule base that mimics the actions the human made. This system runs for a set number of generations before the best individual is selected.
5) The best individual is then placed into the coevolutionary system, replacing the worst individuals from each population, and the whole process iterates again from the beginning.

To ensure the human model affects the coevolutionary populations, the population size was cut down to 15 individuals. Thus, even if the human model has a weaker fitness than the individuals in the population, it still has a probable affect through selection. The first time the coevolution is run, it runs for 300 generations. This is enough to develop a beginner level player that buys some form of weaponry. The consecutive iterations of coevolution then only have 100 generations to coevolve a new player, which allows reasonable time (generally below 1 minute) in between games with the human.

The final component of the process is the human modelling system. This is added through the evolutionary process described in section V. The human modelling system added on average an extra minute to the entire process.

The GUI was coded in Java and communicated to the C++ code through http sockets. Figure 1 shows a screen shot of the GUI mid game.

The environment section shows the current year, the chance of war breaking out at the end of the current year (the *pwar*), the budget for the year, and the amount of the budget left as the user allocates to weapons and INTEL/CI. The previous year's data section shows the total OA, OB, DA and DB utils left over from the previous year (once the opposition's corresponding utils have been subtracted) for both the player, and the opposition. The opposition utils are only shown if INTEL has been purchased, otherwise "UNKNOWN" is displayed. If the opposition has purchased CI, then the value shown in the enemy's previous year's data may be incorrect to some degree. The previous year's INTEL section displays the amount of INTEL bought by the player for the previous year.

The Intelligence Allocation table and Counterintelligence Allocation table both have three columns. Each row in the table represents a different type of INTEL/CI category, with the associated cost. The last column in each table is the user entry field, where the user can enter the budget amount allocated to the category. The cost of INTEL/CI is deliberately low to encourage purchase.

The Weapons table displays all the weapons available for the year. Each row represents a different weapon with the corresponding attributes for the weapon. The first column gives the name of the weapon, consisting of the category (Offensive/Defensive), type (A/B) and number (1,2,3). The second column shows maximum number of weapons that can be acquired (bought) each year. The third column is the cost to purchase a single unit of the weapon. The fourth column represents the inventory for the weapon – the number of weapons given to the player for 'free' at the beginning of the game. The fifth column gives the cost to operate (use) a unit of the weapon for a year. The sixth column gives the number of utils the weapon has (the power ability of the weapon). The seventh and eighth columns show the weapons that have been opted and bought (respectively) in the previous year. Finally, the ninth and tenth columns are the user input columns to allocate the budget to opt previously bought and opted (or available in inventory) weapons, and purchase new weapons for the coming year.

Once a player has made their allocations, the commit button is pressed and play either continues into the next year, or war breaks out and the game results are displayed.

VII. User Study

Using the HACP system, we ran a user study to test the effectiveness with humans. The purpose of the user study was to obtain users with no experience of TEMPO, and use the HACP system as a way of training them. We also wanted to test the effectiveness of using a system that adapts to a human, as opposed to a static player or a coevolving player with no knowledge of the human strategy.

To achieve this experiment, we created an application with three consecutive stages. Each stage would involve the human playing 4 consecutive games against a computer player, with the results for each game recorded. We chose 4 games due to time constraints, but ideally more games would be beneficial. The first stage involved running the user against the same static computer player for all 4 games. The static player was previously evolved and consisted of 19 weapon rules, 8 intelligence rules and 8 counter intelligence rules, with the two most active rules as:

1) IF Category IS Offensive and Type IS B
   THEN Evaluation IS medium
2) IF OperationCost IS Very Low
   THEN Evaluation IS high

The intelligence and counter intelligence rules were rarely activated. After playing the 4 games against this static player, the human was then informed that the next stage was about to start.

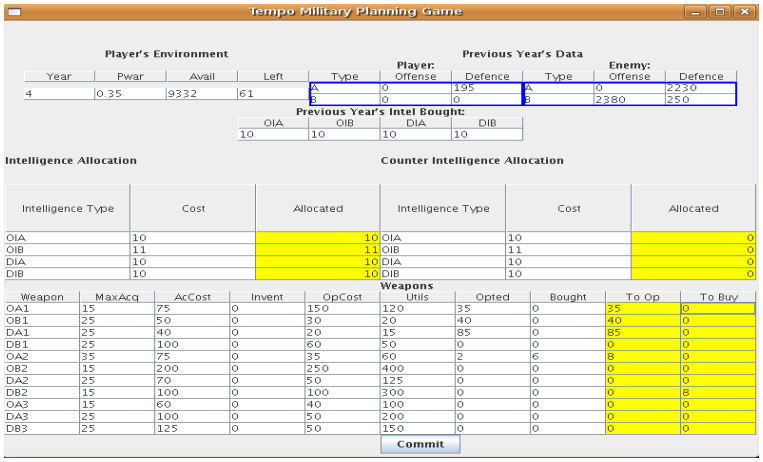

Fig. 1. Screenshot of the TEMPO GUI

Stage 2 consisted of the human playing 4 games against the coevolutionary system. The system was first run for 300 generations (with the same evolutionary parameters as described in section VI), and the individuals for both populations in the final generation were saved. This was the starting point for all the human players, and the best individual from this coevolutionary run was chosen as the starting individual to play against the human for Stage 2. Once the first game was completed, the coevolutionary system was then restarted from where it left off, and another 100 generations were run. The best individual from the best population (according to fitness) was then chosen to play against the human, and this process was then iterated for the rest of the games. While the first coevolved player in Stage 2 was the same for each test subject, consequent players were different due to the coevolution.

Stage 3 was conducted in the same manner as Stage 2. However, in this stage, the HACP system was used, and the additional steps of finding a human model and including it in the coevolutionary system were applied.

A. User study results

The results for Stage 1 are depicted in table I. The table shows the results for each human player, for each game played, against the static computer player. The score is the total net utils for the human player at the end of the game. A positive result shows a win, negative a loss. The number of years the game played for is also recorded for comparison purposes. The final column in the table shows the total number of wins each player had in the stage. The last row in the table gives the average score and years played for each game, and the average games won for Stage 1.

As expected, the results show that the players had an overall average loss for this round. The loss does however decrease over the progression of the stage, depicting player learning. From analysing the results and questioning the players, we found that the average games played before the players felt confident in their game-play were 4–5 games. We also note that there were some players who developed good strategies from Stage 1, and performed well throughout the stages.

The Stage 2 results are depicted in table II, in the same format as the stage one results. By this stage, most of the players begin to develop some winning strategies. There is a dramatic increase in the average scores for the entire stage, although there is still a slight learning curve in some participants. To note, the first game played against is actually a simpler (but different) player than the static one played in Stage 1, so the number of negative scores tends to show some players are still developing their strategies. By the end of this stage we only have a single player that has not won a game, with the majority of players winning at least two games. The players are beginning to play around with strategies, or have found a 'winning' strategy they continue to use.

The Stage 3 results are shown in table II, and follow the same format as before. The first game has the same player as Stage 2, Game 1 (the same coevolutionary starting point), but this time round the majority of players win. At Stage 3, Game 2, the first round of the HACP process has been performed, with the coevolution taking into account the human game-play. The Game 2 results for this stage show an overall success with the HACP system. A large amount of the players who won in Stage 2, Game 2 either lost the game or had the amount of utils won by greatly decreased. The 3rd game in Stage 3 shows this same trend happening with some players, while others succeed with a new winning strategy. The same thing is seen in Stage 3, Game 4.

The other notable thing from Stage 3 is with player 12, who was the only player that had difficulty learning the game. The results from this stage show that even though the losses were continuing, they were not by as much. It appears that even in this case the game-play was slowly helping. Whether this is due to a longer learning curve or the HACP system however is questionable.

VIII. Conclusions and Future Work

The total wins for Stage 3 were slightly higher than Stage 2, which was expected due to the player learning curve

TABLE I

USER STUDY STAGE 1 RESULTS

Stage 1									
	Game 1		Game 2		Game 3		Game 4		
Player No.	Score	Years	Score	Years	Score	Years	Score	Years	Wins
1	-135	4	655	8	0	2	0	3	1
2	-1920	7	1165	2	2480	8	2160	6	3
3	0	3	-4275	8	-2750	4	-50	2	0
4	460	4	-7100	7	2370	8	425	3	3
5	-6661	9	3230	3	-6440	7	-6012	8	1
6	-4340	7	1660	5	0	7	-2000	6	1
7	0	7	-620	5	-1640	5	0	7	0
8	2465	8	2000	5	2610	10	1400	6	4
9	0	7	-1090	5	-99	2	1165	2	1
10	0	8	2713	3	-1290	6	-1550	4	1
11	-1190	6	-3890	7	-1820	7	2730	6	1
12	2260	5	-4820	5	-4640	5	-4940	5	1
Average	**-1029.18**	**6.36**	**-504.73**	**5.27**	**-598.09**	**6.00**	**-157.45**	**4.82**	**1.45**

TABLE II

USER STUDY STAGE 2 RESULTS

Stage 2									
	Game 1		Game 2		Game 3		Game 4		
Player No.	Score	Years	Score	Years	Score	Years	Score	Years	Wins
1	3860	5	-540	4	-495	4	1300	3	2
2	8940	7	4412	8	6589	9	-3004	6	3
3	-2518	7	1200	2	2955	7	-2599	8	2
4	-40	3	6100	7	5039	7	1092	5	3
5	-1155	4	-3358	5	659	6	2940	3	2
6	0	8	3260	6	700	4	0	9	2
7	-1440	3	3070	5	-840	3	0	7	1
8	1065	5	2910	5	615	5	2965	4	4
9	5188	9	1210	7	1278	5	3840	2	4
10	-1236	4	4800	8	289	4	4640	2	3
11	2965	6	4070	5	2604	7	0	3	3
12	-4620	5	-7656	7	-9316	8	-9186	5	0
Average	**1420.82**	**5.55**	**2466.73**	**5.64**	**1763.00**	**5.55**	**1015.82**	**4.73**	**2.64**

TABLE III

USER STUDY STAGE 3 RESULTS

Stage 3									
	Game 1		Game 2		Game 3		Game 4		
Player No.	Score	Years	Score	Years	Score	Years	Score	Years	Wins
1	2785	6	-649	10	-2600	4	0	5	1
2	3960	6	-490	6	-3290	6	2780	6	2
3	-4601	5	-3085	6	730	4	-1899	9	1
4	3765	6	442	9	582	7	2926	8	4
5	47	3	428	3	316	3	-342	4	3
6	-1440	3	3140	4	1640	6	-2090	4	2
7	230	4	1500	6	2148	2	5750	7	4
8	2455	6	3695	7	1019	8	400	3	4
9	4705	8	-586	7	1500	1	110	6	3
10	4995	7	445	7	1500	1	5630	8	4
11	3335	6	2450	6	3210	4	1740	5	4
12	-6418	6	-4894	7	-1180	3	-5050	6	0
Average	**1839.64**	**5.45**	**662.73**	**6.45**	**614.09**	**4.18**	**1364.09**	**5.91**	**2.91**

continuing through into Stage 2. However, the results clearly show that there was not a large jump in results between Stages 2 and 3, and that the system did in fact continue to challenge and teach people. This conclusion was reinforced through the informal verbal feedback process at the end of the user study. The majority of the users stated that they found Stage 3 more challenging than Stage 2 when asked if they noticed any difference between the two stages. There were also some users who specifically stated that strategies they had developed and used in Stages 1 and 2 needed re-evaluation in Stage 3 when the computer player overcame their strategy. The human players were then forced to think of different strategies.

There were also players who were able to dominate the game from Stage 1. This is likely due to the choices made to cater for average users. For example, to allow for an easier starting point, we only evolved the starting computer player for 300 generations. With a population of 15 individuals this only allows creation of a very simplistic player. Additionally,

the short number of games played meant that the HACP system in Stage 3 did not have much time to evolve against more complex human strategies. The human modelling system also struggled more to create these strategies. Even with these deficiencies however, the human players with stronger strategies noted that they thought Stage 3 was beginning to increase in difficulty, and future research should test the impact of additional games.

While the HACP system does seem to have considerable benefit for the majority of users, the extreme ends of the learning curve do not seem to benefit as much. One way to address this problem is to adapt the evolutionary parameters to the capability of the human player. The idea here is to adaptively restrict or encourage the coevolutionary process to match the proficiency of the individual human player.

If a human player is doing particularly poorly and loosing every game, he or she soon feels discouraged and stops playing. Hampering the success of the coevolutionary process can address this issue, be it through a reduction in generations or population size, or applying an additional weight to the evaluation function. The weight could change depending on how much the individual won by against the human player, with the result of allowing 'lesser' individuals to obtain a higher fitness. The human players who continually win against the computer player also need a mechanism to make the play more interesting. The same concept could be applied for these human players, but in reverse. When a computer player loses to a human, the generations and/or population size could adaptively *increase*. The evaluation weighting could be applied to progressively increase the loss penalty for successive losses. There are also many other possibilities that could be applied.

The other area we would like to improve is the development of the human model. Each time a new game is played, a new model is reverse engineered with no reference to previous games. This process wastes a vast amount of data that could be used to refine the model, as usually humans build strategies from previous game-play. One mechanism we have thought of applying is to use the results of the previous year to influence the fitness of the current models being evolved. If a model has some similar tactics to the previous year, then it is rewarded. This concept also has inadequacies however, as it only forms a single link to the previous year, and does not take into account long term strategy.

Overall, we had a great deal of success with the implementation of the HACP system, with a number of users stating that they had fun trying to beat the computer player. Being able to learn new and better strategies of resource allocation on their own time allows students individual training that caters for their own needs. The HACP system provides a good mechanism for applying such training. The HACP system also has benefits for creation of computer players in many commercial games (such as turn based strategy games), where each turn the strategy could be updated to incorporate the human player's strategy. The applications are many, and this research is just the beginning. We have shown that it can work, and making it work in other games is an exciting challenge.

ACKNOWLEDGMENTS

The authors would like to thank all the participants for the user study in this research. Their time and effort were very much appreciated. We would also like to acknowledge MNiSW grant number N516 384734.

REFERENCES

[1] K. Becker, "Teaching with games: the minesweeper and asteroids experience," in *Journal of Computing Sciences in Colleges*, vol. 17, no. 2. USA: Consortium for Computing Sciences in Colleges, 2001, pp. 23 – 33.

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

[3] B. Scott, "AI game programming wisdom," in *The Illusion of Intelligence*, S. Rabin, Ed. Charles River Media, 2002, pp. 16 – 20.

[4] I. L. Davis, "Strategies for strategy game AI," in *AAAI 1999 Spring Symposia*, 1999, pp. 24 – 27.

[5] S. J. Louis and J. McDonnell, "Learning with case-injected genetic algorithms," in *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 4, 2004, pp. 316 – 328.

[6] ——, "Playing to train: Case injected genetic algorithms for strategic computer gaming," in *GECCO*, 2004.

[7] S. J. Louis and C. Miles, "Combining case-based memory with genetic algorithm search for competent game AI," in *ICCBR Workshops*, 2005, pp. 193 – 205.

[8] ——, "Playing to learn: Case-injected genetic algorithms for learning to play computer games," in *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6, 2005, pp. 669 – 681.

[9] C. Miles, S. Louis, N. Cole, and J. McDonnell, "Learning to play like a human: case injected genetic algorithms for strategic computer gaming," in *Congress on Evolutionary Computation*, vol. 2, 2004, pp. 1441 – 1448.

[10] M. Ponsen, H. Muoz-Avila, P. Spronck, and D. Aha, "Automatically generating game tactics via evolutionary learning," in *AI Magazine*, vol. 27, 2006, pp. 75 – 84.

[11] M. Ponsen and P. Spronck, "Improving adaptive game AI with evolutionary learning," in *Computer Games: Artificial Intelligence, Design and Education*, 2004, pp. 389 – 396.

[12] M. Ponsen, P. Spronck, H. Muoz-Avila, and D. Aha, "Knowledge acquisition for adaptive game AI," in *Science of Computer Programming*, vol. 67, no. 1, 2007, pp. 59 – 75.

[13] P. Spronck, "Adaptive game AI," Ph.D. dissertation, Maastricht University, 2005.

[14] P. Spronck, M. Ponsen, I. Sprinkhuizen-Kuyper, and E. Postma, "Adaptive game AI with dynamic scripting," in *Machine Learning*, vol. 63, no. 3, 2006, pp. 217 – 248.

[15] P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma, "Online adaptation of computer game opponent AI," in *Proceedings of the 15th Belgium-Netherlands Conference on Artificial Intelligence*, vol. 2003, 2003, pp. 291 – 298.

[16] D. Aha, M. Molineaux, and M. Ponsen, "Learning to win: Case-based plan selection in a real-time strategy game," in *Case-Based Research and Development*, vol. 3620/2005, 2005, pp. 5 – 20.

[17] P. Funes, E. Sklar, H. Juillé, and J. Pollack, "Animal-animat coevolution: Using the animal population as fitness function," in *Animals to Animats 5: Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior*. MIT Press, 1998, pp. 525 – 533.

[18] P. Funes and J. Pollack, "Measuring progress in coevolutionary competition," in *Animals to Animats 6: Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior*. MIT Press, 2000, pp. 450 – 459.

[19] P. Avery, Z. Michalewicz, and M. Schmidt, "Short and long term memory in coevolution," in *International Journal of Information Technology and Intelligent Computing*, vol. 3, no. 1, 2008.