Developing Artificial Intelligence Agents for a Turn-Based Imperfect Information Game

Wilfrido Perez Cutright

A Senior Thesis submitted in partial fulfillment
of the requirements for graduation
in the Honors Program
Liberty University
Spring 2019

Acceptance of Senior Honors Thesis

This Senior Honors Thesis is accepted in partial fulfillment of the requirements for graduation from the Honors Program of Liberty University.

_____

Melesa Poole, Ph.D.
Thesis Chair

_____

Mark Merry, Ph.D.
Committee Member

_____

Scott Long, Ph.D.
Committee Member

_____

David Schweitzer, Ph.D.
Assistant Honors Director

_____

Date

Abstract

Artificial intelligence (AI) is often employed to play games, whether to entertain human opponents, devise and test strategies, or obtain other analytical data. Games with hidden information require specific approaches by the player. As a result, the AI must be equipped with methods of operating without certain important pieces of information while being aware of the resulting potential dangers. The computer game GNaT was designed as a testbed for AI strategies dealing specifically with imperfect information. Its development and functionality are described, and the results of testing several strategies through AI agents are discussed.

Developing Artificial Intelligence Agents for a Turn-Based Imperfect Information Game

Introduction

Since the 20th century, when John von Neumann and Oskar Morgenstern (1944) formally expressed their Theory of Games, people have increasingly viewed the process of decision-making in mathematical terms. The unprecedented rate of the proliferation of computers and the exponential increase in capabilities that have occurred over the same timespan have acted as a much-appreciated catalyst for further research in this area. Not only are the devices used to sift through unfathomable quantities of data to detect patterns that may reveal facets of the human decision process, but the drive to automate as much as possible has led to myriad systems where computers themselves are responsible for decisions, a phenomenon known as artificial intelligence (AI). Combining in this way the logic and planning techniques that human minds employ on a regular basis with the raw computational power and nearly immeasurable information available to computers today has often been successful in the past and will likely continue to bring advances in multitudinous fields of human knowledge (Buchanan, 2005). In accomplishing such feats, the use of concepts from game theory is effective in guiding the decision-making process of the computer. This, in turn, involves identifying the problem that the computer is trying to solve and, if necessary, breaking it into component problems. The intention of such an analysis is to match the issues with scenarios where solutions can be tested and optimal approaches determined. The application of concepts from game theory forms the basis for AI, and the ability of computers to simulate such situations leads to further expansion of the theory.

Problems explored in game theory include those that players of nondeterministic games face when making decisions. Hence, conclusions drawn from the analysis of players' decisions in these types of games, where there are significant stochastic elements, can potentially be used to inform answers to the larger questions posed by the problems themselves. This is the principle behind the research that forms the body of this thesis. Taking advantage of the modeling capabilities of computers, GNaT, a virtual game that presented the players with a combination of several types of problems, was developed by the author. Imperfect information, a type of complication that precludes players from knowing certain pieces of information and thus negatively affects their ability to accomplish their goals, was the foremost issue posed to players within the game. GNaT's integrated AI, implemented in the form of modules that behaved in accordance with the selected strategies for their roles, was consequently tailored to deal with imperfect information and other difficulties using methods reliant upon probability within the game. The respective efficacies of the modules in answering the given problems were experimentally evaluated, resulting in an agent comprised of the relatively best-performing strategies for each task.

Literature Review

The application of artificial intelligence to games is a common exercise and has been fruitful in determining decision-making strategies for those games. While not all AI agents developed for games are suitable for all types of games, various AI algorithms have been devised to deal specifically with common issues in many games such as imperfect information.

Notable Approaches to the Issue of Imperfect Information

Information Set Monte Carlo Tree Search is a member of the family of Monte Carlo Tree Search algorithms. While it shares the same general stochastic process characteristic of Monte Carlo algorithms, it uses search trees comprised of information sets of game states. It has been successfully applied to card games such as Spades and Scopone (Di Palma & Lanzi, 2018). Multiple members of the counterfactual regret minimization family of algorithms have seen great success playing several variants of poker (Brown & Sandholm, 2018). Heuristic-based strategies can also be used for AI. In zero-sum games, these can be implemented in a straightforward fashion if the Nash equilibrium is known. (The Nash equilibrium describes the state in a game where all players are aware that they cannot improve their standings by changing solely their own strategy.) GNaT relies heavily on elements from the two-player hand game Rock-paper-scissors, whose Nash equilibrium is the state where both players randomly play each option with equal probability (van den Nouweland, 2007).

Procedure

The development of GNaT primarily served to facilitate exploration of AI strategies when operating in contexts with incomplete information. AI agents were designed to test various strategies for handling problems presented within the game, chief of which were incomplete information and shifting probabilities.

Description of the Game

GNaT is a zero-sum turn-based game designed for two players. The objective of the game is to eliminate the opponent by reducing his or her health to zero. This is

accomplished by performing “attacks”, which are actions that require certain amounts of in-game currency. Each attack can be accompanied by one card, played from the player’s hand, that assigns an “attack type” – rock, paper, or scissors. If the attack type is advantageous against the opponent’s “defense type” (using the relationships defined by the popular hand game Rock-paper-scissors), the opponent loses a larger amount of health. Less health is lost by the opponent if the attack type is disadvantageous.

The primary mechanic that drives players’ turns is the roll, which randomly selects one of six actions for the player to perform. Each of these is assigned a certain weight. One of the actions available to players is the choice of adding to one of the weights; in this way, the probabilities upon which the roll depends can be altered. Hence, if a player optimizes the probabilities according to their preferences, they can perform desired actions more frequently.

The key to winning the game lies in mastering both of these mechanics. Correctly predicting the defense type of the opponent will most efficiently make use of opportunities to attack, and adjusting the probabilities of certain actions may reduce the number of actions taken in achieving the goal.

Development goals for the game. GNaT was written in C++ using Microsoft Visual Studio Community 2017. The language was chosen for its object-oriented features that allowed clear definitions of classes and their interactions. Since the game was to serve as a test case for AI strategies, simplicity was a stated design goal. This restricted the scope of the project and prevented the addition of features that would have increased its complexity. This consequently meant human enjoyment was not the main focus;

rather, the intention was to give some level of clarity in comparing approaches to AI for the game.

Design elements of the game. Using the principles conceived during early stages of development as a basis, the design of GNaT was conscious and fairly calculated; the end product reflects this. The foremost example of this is the centrality of AI, which manifests itself within the program in several ways. Upon running the executable, the game opens in the console, posing the following question: “Do you want to play against the computer, or do you want to pit two AIs against each other?” The logic used by AI agents is integrated into the Player class, and the game produces lengthy scripts detailing their movements. Information hiding is another underlying principle of the game’s design; access to information about the players is heavily controlled. Players’ defense types can only be revealed to their opponents through the purchase of attacks, and the contents of players’ hands and customized probability weight sets remain private for the entire duration of the match. This restriction of knowledge serves as the basis for the problem of incomplete information within the game, in contrast to Rock-paper-scissors, where information about the opponent is unknown due to the simultaneous actions of players.

Description of AI Agents

The AI were designed shortly after the game concept was solidified. Rather than initially being designed as whole agents, the AI consisted of individual modules that dealt with making decisions at each point where the game required input. Some of these modules were explicitly related; one strategy for modifying the weights of actions in an

automated player's probability set directly references the response of another module designed for selecting actions on a turn. However, for the most part the modules were approached as logically independent segments of the program, each concerned with separate tasks. This led to a search for a combination of individual modules that functioned best in general, rather than a more cohesive one comprised of parts that all conformed to a single unified strategy.

Strategies of each module. These modules interact with the game at the five points where input is required from the player. Each module is the implementation of a strategy for determining which response to input at a particular point.

Purchasing attacks. This first point requiring player input occurs when the player rolls to "visit the shop" and gains the opportunity to purchase attacks. There are three levels of attack; the more a player pays for the attack, the greater value they will receive for their currency. Players also have the option to not purchase an attack at all.

Three strategies were developed for this decision point. The first, given the moniker *Spend!*, dictates simply that the player should always spend at the highest level they can afford at the moment. The second strategy attempts to use the player's knowledge of the opponent's defense type; it prescribes the hoarding of funds until either the elimination of the opponent is guaranteed by an attack or the accumulation of surplus currency begins, in which case attacks with the intention of discovering the opponent's defense type will be made. This behavior earned it the name *Save up until kill*. Finally, the third strategy, named *Go for the double!* consistently attempts to make advantageous attacks through knowledge of the opponent's defense type; if the opponent's defense type

is unknown and the player has at least enough currency to purchase the lowest-priced attack twice, the lowest-priced attack will be purchased, revealing the defense type. The hope is that the player will receive a chance to attack again on a subsequent turn before the opponent can change his or her defense type.

Playing cards with purchased attacks. Playing cards alongside attacks can be risky, paying off with double damage if the type is advantageous or halving damage when the type is at disadvantage. Not playing a card results in a loss of 50 power, regardless of the attack purchased. Even so, only one strategy was developed for this decision point as it likely anticipates every situation. In summary, it seeks to always maximize the damage done; if the opponent's defense type is unknown, the AI agent will guess using the most common type of card the player's hand contains. The exception to this is the rare case when the opponent's health is low enough that elimination is guaranteed when not playing an attack card and the opponent's defense type is unknown, in which case guessing with an attack card is avoided and the kill secured.

Choosing a defense type. Two approaches were developed for making a decision on the player's defense type. Both are consistent with the Nash equilibrium for Rock-paper-scissors: a mixed strategy of equal probability of choosing any of the three options. The sole difference between the first strategy, *Random defense!*, and the second, *Pick defense least likely to be doubled*, does not come into play until the player has acquired a certain number of the cards (4/15 of the deck). The rationale for this is that the deck is finite; this second approach anticipates that if a significant portion of the deck is in the player's hand, the type of attack card of which they have the most is less likely to be

played by the opponent. Hence, it dictates that the type that is at a disadvantage against the most common type of card in their hand should be chosen as the defense type.

Selecting an action. While significantly less likely, it is possible to roll the action called *select*, which enables the player to choose from any of the other actions that can be rolled. The sole strategy for selecting the action uses a function labeled *determineGameState*, which attempts to determine the most urgent action through a series of if statements that assign weights to each choice. The final decision is then determined randomly; however, some weights assigned in certain cases, such as when the opponent is almost eliminated, can dominate in such a way that the options given those weights are nearly guaranteed.

Raising the probability of an action being rolled. Changing the weights of actions that can be rolled is GNaT's mechanism for allowing the player to determine which actions he or she would like to potentially perform more often. The initial weights for each action have values of six with the exceptions of the weight for the action of adding to weights, which has a value of four, and the weight for the player voluntarily selecting an action, which has a value of two. Additionally, when the value of each weight is raised, it is raised by two, aside from those values representing the two actions that have lower initial values; they are raised by one. The pertinent question, then, regards which action will provide the most utility when the value of its weight is raised. A human player would likely find *select* to be the action of most utility, but as its value can only be raised at a lower rate, raising the value of its weight may not be the most beneficial option. The first strategy, *Always choose select*, assumes *select* remains the

best general option despite the lower rate of increase of its weight value. The second, *Select OR raise probability OR use game state*, similarly maintains that *select* is a good general option, but also consults the function *determineGameState* to take its analysis of the current game's state into consideration. To a lesser extent, this strategy also considers the action of raising a probability. The weight whose value is to be raised is then chosen randomly from these three alternatives.

Experimental Testing of the AI Strategies

To determine the relative utility of these strategies, combinations of AI modules were constructed. When two separate modules for a given decision point were to be contrasted, they would be put in combinations of modules that were otherwise identical. Then the two combinations would each be assigned to a player within the game, and the game would run until one of the two players met the win condition. Execution would be repeated for numerous iterations. The starting player (randomly decided at the beginning of the game), total number of turns taken, and the state of each player's health, currency, and number of cards at the end of the game would be recorded. These data served as diagnostics for determining which of the two different modules worked better with that combination of modules. Since only three of the decision points had more than one strategy, only modules from those points needed to be tested against each other.

Tests performed. The first two modules to be tested corresponded to the action of purchasing an attack: *Spend!* and *Save up until kill*. The combination of modules with which they were each paired was comprised of the sole strategies for the decision points for playing cards and selecting an action, the *Random defense!* strategy for choosing a

defense type, and the *Always choose select* strategy for determining which weight values to raise. (Henceforth, the modules that are the only strategies for their respective decision points will be omitted when listing the combinations of modules used for testing.) The two AI agents were run against each other 16 times, with the AI using the *Spend!* module designated Player 1 and the one using *Save up until kill* designated Player 2. The results are shown in Table 1.

Table 1

Spend! (Player 1) v. Save up until kill (Player 2)

	Wins	Average turns in games won	Average health during win
Player 1	8	64.6	931
Player 2	8	83.4	503

Retaining the same combinations of modules for each player, with the exception of replacing *Save up until kill* with the *Go for the double!* module for Player 2, the two players were run against each other another 16 times, obtaining the results in Table 2.

Table 2

Spend! (Player 1) v. Go for the double! (Player 2)

	Wins	Average turns in games won	Average health during win
Player 1	11	74.7	532
Player 2	5	73.0	325

Following these tests, the two modules for the decision point associated with choosing a defense type were each placed into a combination containing *Spend!* and *Always choose select*. Player 1 utilized *Random defense!*, while Player 2 instead used the *Pick defense least likely to be doubled* module. The two players faced off 20 times, leading to the outcomes specified in Table 3.

Table 3

Random defense! (Player 1) v. Pick defense least likely to be doubled (Player 2)

	Wins	Average turns in games won	Average health during win
Player 1	10	74.9	603
Player 2	10	76.3	505

For the final decision point, each player's combination of modules included *Spend!* and *Random defense!*. Player 1 also included *Always choose select*, while Player 2 featured the *Select OR raise probability OR use game state* module. The players dueled for 20 matches, as shown in Table 4.

Table 4

Always choose select (Player 1) v. Select OR raise probability OR use game state (Player 2)

	Wins	Average turns in games won	Average health during win
Player 1	10	75.9	540
Player 2	10	80.5	575

Finally, a test to determine whether there exists a significant advantage or disadvantage as a result of starting the game was conducted. The initial assumption when developing the game was that there would be an advantage, so the player to play second was given 100 additional health as a counteractive measure. Both players in this scenario were formed from the same modules, namely, *Spend!*, *Random defense!* and *Always choose select*. They opposed each other 20 times; the results are documented in Table 5.

Table 5

Starting Player v. Second Player

	Wins	Avg. turns in games won	Avg. health during win
Starting Player	10	81.0	565
Second Player	10	74.4	635

Conclusion

Performance Evaluation of the AI Agents

The simplest way to evaluate the utility of the strategies used by the AI agents is by comparing their performances when pitted against each other. Given enough trials, any substantial correlations in the data should become apparent.

For the first decision point, two of the strategies seemed competitive, while the third did not compare well to one of the other two. *Spend!* and *Save up until kill* performed similarly, though the former seemed to have a slight advantage judging by the average turns and average remaining health during Player 1's wins. When *Spend!* and *Go*

for the double! competed, the results showed a significant discrepancy in their abilities to defeat their respective opponent.

The two strategies for the decision point for choosing a defense type performed similarly, although *Random defense!* may have had a slight advantage given that the player using the module, on average, won with more health in fewer turns. It was expected that they would experience similar results, as they both use the same basic strategy until a certain point in the game is reached. The condition specified by *Pick defense least likely to be doubled*, dependent upon the number of cards in that player's hand, may have prevented the strategy's unique code from executing in shorter matches.

Even more evenly matched were the modules used for raising the probabilities of particular actions. The slightness of the differences in their average remaining health and average game lengths may indicate that they perform at equivalent levels; alternatively, it may suggest that changing the probability of various actions does not matter much in the outcome of games played by the AI.

The test to determine whether the starting player has an advantage did not result in large differences between outcomes for the starting player and second player. If one has an advantage, the data indicate that it would be the player who plays second, possibly due to the extra health they receive when turn order is decided.

Implications

Incomplete information is a common factor in conflicts between opposing parties. As a result, methods of operating despite this lack of knowledge are often essential to defeating an opponent. Since various approaches exist for dealing with hidden

information, ascertaining the superior one for a given problem is desirable. GNaT is a game that models the problem of incomplete information in order to test the efficacy of AI approaches. The performances of the different AI strategies within the game reflect their success in mitigating problems imposed by the game. Hence the efficacy of their respective methods can be analyzed with respect to the particular types of problems they were developed to face. Conclusions drawn from such an analysis may prove useful in solving similar problems in other contexts.

Limitations

The primary limitation for this study was the lack of computing power necessary to implement advanced algorithms such as those of the Monte Carlo Search Tree family, which generate trees of possibilities, following randomly selected nodes to their endgame results before committing to particular actions. More computers would also allow for tests to be run thousands or possibly millions of times, achieving a more accurate picture of trends within the data.

Further Research

Many aspects of GNaT remain to be explored. For instance, the function *determineGameState* is an amalgamation of tests for various conditions. Experimenting with the combinations of if statements contained within the function may lead to developing more accurate assessments of the game's state. Adding additional diagnostics to tests may reveal further correlations between the performance of AI modules and variables such as game length, the specific actions whose probabilities of being rolled were increased, and player hand size. If more computational power is acquired, machine

learning techniques can be applied to possibly determine entirely new strategies. GNaT can also be modified to allow for new elements of gameplay to be tested. It is probable that further efforts such as these will be fruitful, contributing to a better understanding of AI in games.

References

Baier, H., Sattaur, A., Powley, E. J., Rollason, J., & Cowling, P. I. (2018). Emulating human play in a leading mobile card game. *IEEE Transactions on Games.* https://doi.org/10.1109/TG.2018.2835764

Bakkes, S., Spronck, P., & van den Herik, J. (2009). Rapid and reliable adaptation of video game AI. *IEEE Transactions on Computational Intelligence and AI in Games, 1*(2), 93-104. https://doi.org/10.1109/TCIAIG.2009.2029084

Bakkes, S. C. J., Spronck, P. H. M., & Lankveld, G. (2012). Player behavioural modelling for video games. *Entertainment Computing, 3,* 71-79. https://doi.org/10.1016/j.entcom.2011.12.001

Bitan, M., Kraus, S. (2017). Combining prediction of human decisions with ISMCTS in imperfect information games. Retrieved from https://arxiv.org/abs/1709.09451

Brown, N., & Sandholm, T. (2018). Solving imperfect-information games via discounted regret minimization. Retrieved from https://arxiv.org/abs/1809.04040

Brown, N., Sandholm, T., & Amos, B. (2018). Depth-limited solving for imperfect-information games. Retrieved from https://www.researchgate.net/publication/325282968_Depth-Limited_Solving_for_Imperfect-Information_Games

Buchanan, B. G. (2005). A (very) brief history of artificial intelligence. *AI Magazine, 26*(4), 53-60. Retrieved from http://ezproxy.liberty.edu/login?url=https://search.proquest.com/docview/208132026?accountid=12085

Chitizadeh, A., & Thielscher, M. (2018). Iterative tree search in general game playing with incomplete information. Retrieved from https://www.semanticscholar.org/paper/Iterative-Tree-Search-in-General-Game-Playing-with-Chitizadeh-Thielscher/4e6835257f7ca348ab35925ed18546c20b1ebe9b

Di Palma, S., & Lanzi, P. L. (2018). Traditional wisdom and Monte Carlo tree search face-to-face in the card game Scopone. *IEEE Transactions on Games*, *10*(3), 317-332. https://doi.org/10.1109/TG.2018.2834618

García-Sánchez, P., Tonda, A., Mora, A. M., Squillero, G., Merelo, J. J. (2018). Automated playtesting in collectible card games using evolutionary algorithms: A case study in Hearthstone. *Knowledge-Based Systems, 153*, 133-146. https://doi.org/10.1016/j.knosys.2018.04.030

Heinrich, J. (2016). Reinforcement learning from self-play in imperfect-information games. Retrieved from https://arxiv.org/abs/1603.01121

Hom, V., & Marks, J. (2007). Automatic design of balanced board games. *Proceedings of the Third Artificial Intelligence and Interactive Digital Entertainment Conference*, 25-30. Retrieved from https://aaai.org/Library/AIIDE/aiide07contents.php

Justesen, N., Mahlmann, T., Risi, S., & Togelius, J. (2018). Playing multiaction adversarial games: Online evolutionary planning versus tree search. *IEEE Transactions on Games, 10*(3), 281 - 291. https://doi.org/10.1109/TCIAIG.2017.2738156

Kitchen, A. C., & Benedetti, M. (2018). ExIt-OOS: Towards learning from planning in imperfect information games. Retrieved from https://arxiv.org/abs/1808.10120

Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Perolat, J., . . . Graepel, T. (2017). A unified game-theoretic approach to multiagent reinforcement learning. Retrieved from https://arxiv.org/abs/1711.00832

Mark, D. (2009). *Behavioral mathematics for game AI*. Boston, MA: Course Technology PTR.

Omarov, T., Aslam, H., Brown, J. A., & Reading, E. (2018). Monte Carlo tree search for Love Letter. Retrieved from https://www.researchgate.net/publication/327679828_Monte_Carlo_Tree_Search_for_Love_Letter

Powley, E. J., Cowling, P. I., & Whitehouse, D. (2017). Memory bounded Monte Carlo tree search. Retrieved from https://aaai.org/ocs/index.php/AIIDE/AIIDE17/paper/view/15856

Sun, Q., & Ganzfried, S. (2018). Bayesian opponent exploitation in imperfect-information games. https://doi.org/10.1109/CIG.2018.8490452

Takaoka, Y., Takashi, K., & Ooe, R. (2018). A fundamental study of a computer player giving fun to the opponent. *Journal of Computer and Communications, 6*(1), 32-41. https://doi.org/10.4236/jcc.2018.61004

van den Nouweland, A. (2007). Rock-Paper-Scissors; a new and elegant proof. Retrieved from https://minerva-

access.unimelb.edu.au/bitstream/handle/11343/34714/67348_00003579_01_1003.pdf?sequence=1&isAllowed=y

von Neumann, J., & Morgenstern, O. (1944). *Theory of games and economic behavior*. Princeton, NJ: Princeton University Press.

Zhang, S. (2017). Improving collectible card game AI with heuristic search and machine learning techniques. https://doi.org/10.7939/R34X54W6H