Intelligent Players Competing in the Game of Qubic

James G. Schiiller, Greg L. Vitko, and Gautam B. Singh
Department of Computer Science, Oakland University, Rochester, MI, United States

Abstract - *The objective of this research is to develop two intelligent players to competitively play against each other in the game of Qubic. The two-player computerized board game environment possesses an excellent setting in which to conduct programming experiments. One player selected to compete in the contest is a deterministic, heuristic-based approach to artificial intelligence that plays well. The other player selected is a non-deterministic, genetic programming-based approach to machine learning. The immediate results clearly show the heuristic player as the dominant one. Two-pass evolution is applied to strengthen the genetic programmer well enough to win more than a few matches against the heuristic opponent. The analysis opened up areas of research into developing stronger intelligent heuristic and genetic programming players.*

Keywords: Artificial Intelligence in Games, Genetic Programming, Heuristics, Qubic

1 Introduction

The two-player computerized board game environment possesses an excellent setting in which to conduct programming experiments. The ultimate goal is to establish one winner for each game played. The outcome of the games, in their entirety, holds the key-point in which to compare intelligent players. Qubic, an advanced version of the traditional tic-tac-toe game, is the board game chosen for this study. The research trials were conducted with a two-player stance, competing over a 100 game set of Qubic. One of the competing players is the Open-123 heuristic; the other competing player is referred to as genetic programming (GP). The object of this research paper is to: develop two intelligent Qubic players; compete the players against each other; discover the final results of each game played; examine the moves; interpret the results of the analysis and state the final verdict of the project. The conclusion of the experiment will establish that the player with the most wins at the end of the tournament is the ultimate winner of the competition.

Keep in mind, there are more properties of the competition to consider than just the final outcome. These properties should not be dismissed as inconsequential. Although the outcome is decisive, many other factors are equally as important, since they may hold considerable bearing on the eventual outcome. For instance, the winner could move at an extremely slow pace while the loser moves more swiftly. Additionally, the loser may have an interesting playing style or possesses the potential to play at a higher-level, and so on. The ideas discovered while playing Qubic can be mapped to less trivial and more serious problems, e.g.) the medical field. Games could be the domain where "it" all starts).

2 Qubic

2.1 The Game of Qubic

Qubic is a board game originally sold by Parker Brothers, in the late 1960's. Qubic consists of four 4x4 tic-tac-toe boards. Qubic is easier to understand, visualize, and talk about in a two-dimensional representation as seen in Figure 1. Additionally, the boards can be combined together to form a three-dimensional cube as seen in the same Figure 1. [1]

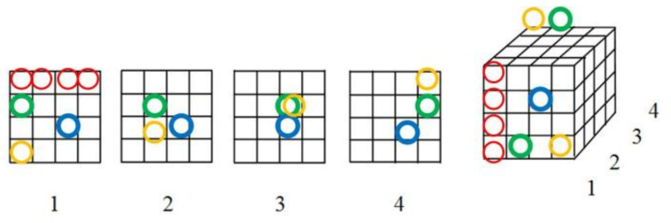


Figure 1: Two- and three-dimensional Qubic in layers.

Each board layer labeled 1, 2, 3, 4 are the same as the corresponding two-dimensional board representation. Sixteen squares on each of the four boards total the entire 64 square playing surface. The object of the game is to line four "X's" or "O's" in a row, along, for instance, the horizontal, vertical, or diagonal axes, on one square of each board, beginning with an outer board. X is the first player to move. Players then alternate moves. The first player to line four of their squares up in a row wins.

2.2 Qubic Mathematics

There are 76 winning lines in Qubic. Further details are contained within [2]-[4]. See Table 1 to view all of the 76 possible winning move sets. Qubic has an upper bound complexity level of 3^{64}. Each square can be filled by either X, O, or empty [5]. There are 64 total squares. That is, multiply three times three, 64 times. The result is an astronomical number. There are this many combinations (total number of

Table 1: The 76 winning line sets in Qubic

Vertical	Horizontal	Straight up	Vertical / Horizontal	Diagonal
{0,1,2,3}	{0,5,10,15}	{0,20,40,60}	{0,21,42,63}	{0,6,12,18}
{5,6,7,8}	{1,6,11,16}	{1,21,41,61}	{5,26,47,68}	{3,7,11,15}
{10,11,12,13}	{2,7,12,17}	{2,22,42,62}	{10,31,52,73}	{20,26,32,38}
{15,16,17,18}	{3,8,13,18}	{3,23,43,63}	{15,36,57,78}	{23,27,31,35}
{20,21,22,23}	{20,25,30,35}	{5,25,45,65}	{0,25,50,75}	{40,46,52,58}
{25,26,27,28}	{21,26,31,36}	{6,26,46,66}	{1,26,51,76}	{43,47,51,55}
{30,31,32,33}	{22,27,32,37}	{7,27,47,67}	{2,27,52,77}	{60,66,72,78}
{35,36,37,38}	{23,28,33,38}	{8,28,48,68}	{3,28,53,78}	{63,67,71,75}
{40,41,42,43}	{40,45,50,55}	{10,30,50,70}	{3,22,41,60}	{0,26,52,78}
{45,46,47,48}	{41,46,51,56}	{11,31,51,71}	{8,27,46,65}	{15,31,47,63}
{50,51,52,53}	{42,47,52,57}	{12,32,52,72}	{13,32,51,70}	{3,27,51,75}
{55,56,57,58}	{43,48,53,58}	{13,33,53,73}	{18,37,56,75}	{18,32,46,60}
{60,61,62,63}	{60,65,70,75}	{15,35,55,75}	{15,30,45,60}	
{65,66,67,68}	{61,66,71,76}	{16,36,56,76}	{16,31,46,61}	
{70,71,72,73}	{62,67,72,77}	{17,37,57,77}	{17,32,47,62}	
{75,76,77,78}	{63,68,73,78}	{18,38,58,78}	{18,33,48,63}	

positions) in Qubic. If you wanted to get an even more accurate picture of the complexity of Qubic, use 64 factorial, an even larger number, as the upper bound complexity level estimate of Qubic [6]. This estimate includes the 364 estimate plus it counts move order, a key game property to consider [6]. Counting move order is the difference between the two estimates. Either way, it means the Qubic search space is extremely large. This qualifies it as a computationally intractable game (NP-hard problem, concerning the time required to solve the game). We are forced to estimate solutions to the problem.

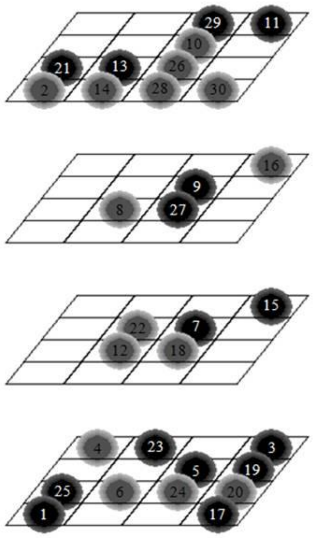
3 Intelligent Players

Processes found within the gaming communities are integral to the development of artificial intelligence (AI). Intelligent players are AI in the form of computer programs developed to play board games. Intelligent players have the notable ability to quickly solve complex problems. At the same time, intelligent players have trouble solving simple, very basic (common sense) types of problems. Two AI players can complete nearly 100,000 games of Qubic in a matter of hours. The same amount of games would take two human players, 100,000 consecutive minutes (or two months) of play to complete at the alarming rate of one minute per game.

3.1 Heuristics

Heuristics are deterministic, problem-dependent, and rule-based solutions to problems. Human programmers consult with experts in the field to take human experiences and advice, trial and error, educated guesses, or even brute force tactics, and transform the knowledge into computer procedures and programs. Improvement to the heuristic requires human programmer intervention. The problem with heuristics is the knowledge is considered shallow and the expertise is limited to the domain the system knows about. Heuristics do not generalize their knowledge. However, heuristics can provide valuable shortcuts that can reduce both time and cost. [7]

The heuristic player is one of the players to compete in the Qubic contest. The power of the heuristic player is contained within the evaluation function. The logic behind the evaluation function is called Open-123. The heuristic player is deterministic; it will make the same exact move each time it is given the same situation. The logic defining the evaluation function is simple. Higher values are assigned to moves containing a higher number of friendly squares, meaning squares filled by the player it is evaluating. Each line in each square is summed to give one total value to the square. Open 3 is assigned a higher value than Open 1. If there is a square filled by an enemy in the line being evaluated, the value is automatically returned as 0.0. The square containing the highest value is returned from the evaluation function. This is the square it will move to on the actual board. The Open-123 evaluation function can be used in opposites as the player's defense logic to block. Before evaluating itself, it can evaluate the opponent in order to detect an immediate win threat. Then, according to the killer heuristic philosophy, move to this very spot first before the opponent. During development, the heuristic player was benchmarked against the skill level of human players. It was found to be extremely difficult for humans to calculate what square Open-123 will play on next. The player was good enough at the moment in time when the author of this work and other subjects were unable to defeat it. One person who was able to beat the player was Greg Vitko (Chess master). Vitko notably achieved the win in 30 moves on the first attempt, while playing second at a disadvantage. The other human players did not do as well. The final verdict was the Open-123 player plays well. There were some weaknesses exposed. All around, it evaluates well enough to serve its purpose. The move string and graphic game of the winning sequence can be seen in Figure 2. The graphical example game on the tilted boards below are the graphics developed by David Cao and reused to clearly demonstrate example games of Qubic.



0,60,18,03,12,6,32,46,52,72,78,26,66,65,38,58,15,31,17,16,61,27,8,11,1,71,51,70,73,75

Figure 2: Heuristic player defeated.

3.2 Genetic Programming

Genetic programming is a new, resurrected, exciting, and actively being researched area. It has been around since the 1960's and taken on and off of the shelf during the 1980's. It has been shelved in the past because of the limitations on computing power and heavy requirements placed on execution. Any modern personal computer satisfies the power requirements. Genetic programming is being revisited today.

Genetic programming is a machine learning technique inspired by the theory of biological evolution. Genetic programmers are algorithms that create algorithms. In tree-based GP, the generated computer programs are similar to decision trees and more precisely parse trees: decisions travel up the tree. Each node can be a complex mathematical formula (function) [8]. The general idea is to first try a few thousand random guesses to see which ones are best; then, take the random guesses and improve them. The basic work flow follows. First, generate the initial population of random programs built from the primitive function set. Then, rank the programs by evaluating them in some predefined way. The best programs are then selected. These are mutated and bred with crossover (recombination). The process is continued until the termination condition is satisfied.

GP programs in the Qubic implementation are stored as tree structures. The trees are initialized with a primitive set of five functions shown in the following list.

1) add(l):=return $l_0 + l_1$
2) multiply(l):=return $l_0 * l_1$
3) subtract(l):=return $l_0 - l_1$
4) if(l):=if $l_0 > 0$:return l_1 else:return l_2
5) is_greater(l):=if $l_0 > l_1$:return 1 else:return 0

Each function contains a set of simple operators. The GP recursively recombines and forms these functions into programs. The nodes on the trees are function, parameter, or constant nodes. For each function node there is a list *l* of other function, parameter, and constant nodes. The functions are represented in the list; the parameters are simple auto-generated lookup values determined at runtime; and the constants are hard coded integer values.

Trees are programs. They calculate their next move by propagating return values up the tree from the bottom nodes, towards the root of the tree, formulating to the output number in which the GP uses to make its next move (see Figure 3 for an example of a genetic program). Programs are designed by means of evolutionary pressure. The best player remains or is replaced, dependent on how well it plays against other competitors in its run. To be useful, the GP should be serialized to disk. The player can then be brought back to life and compete in more rounds of evolution and serialized back to disk again.

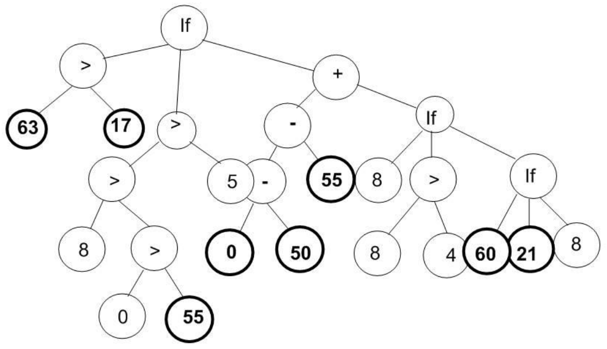


Figure 3: Example tree-based program.

4 Experiment

4.1 Hypothesis

The purpose of this experiment is to compete two AI players against each other in a 100 game contest of Qubic, to see who wins and to see what thoughts emerge. Solving the game of Qubic is not the purpose. Qubic has already been weakly solved. Qubic is not being used for that purpose, but it supplies the domain which conducted the comparison results. It is the hope of this study to develop the GP player to win at least a few games against the heuristic player. If this experiment reveals the GP loses, it means there are open areas for future work.

4.2 Experiment Results

The heuristic player won every single game! The length of the games is an indicator of the equality of the players. This is true when humans compete against other human players in games such as chess. It is logical to deduce the same for AI players for Qubic. An average game of Qubic with roughly equal players lasts 20 moves. In this contest, an average game lasted only 10 moves. The length is not long enough to warrant equality of the players.

5 Opened Areas of Research

5.1 First Attempt Failed

The results of the experiment opened up areas of research into developing stronger intelligent GP players. The basic problem is the GP player lost all 100 games. This opened up an opportunity to improve the evolutionary process of the genetic programmer. There were three main problems with the earlier approach:

1) GP was not blocking immediate win threats (basic need).
2) The is_greater primitive function was returning 1/0.
3) The evolution process only involved self-competition.

5.2 Second Attempt Plan

In the first contest, the outcome of running the 100 game set of Qubic resulted in the heuristic player winning every single game. This means one pass of 100,000 evolutions of the GP player results in a poor playing program. Increasing the game experience would not improve the player. The genetic programmer is not executing basic needs by failing to block simple initial 'X' player moves 0, 18, 12 as seen in Figure 4. Computers are not skilled at handling common sense type of basic problems. The first problem to solve on the way to beating the heuristic player is to endow the GP player with basic logic to block. The GP needs to defend against immediate win threats. The second problem to address is to modify the is_greater primitive function to return moves, ranging from 0 - 78, including a padding of bits separating each 4x4 board, instead of returning binary numbers. Additionally, to address the third problem, instead of solely competing against itself, let it compete against a predator opponent.

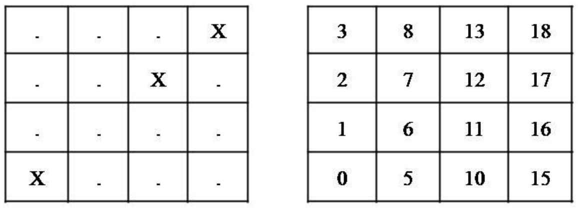


Figure 4: Genetic programmer not blocking.

5.3 Solutions

The solution to the first main problem, to block, is apparent, and easy to correct. Use the same Open-123 logic as the heuristic player for defensive moves. The solution to the second problem requires more thought and includes the blocking solution for the first problem. After much experimentation, it was found that the set of initial programs to start the GP out with should be geared more towards Qubic. One specialized function was made to reference an in-memory lookup table into a small subset of winning lines. This idea fits into the way humans think. During a game of Qubic, human players follow six or seven solution lines they know will win. If one of the lines is blocked, they try another line. People also combine solutions to collect threats. The new function content pulls solutions from the 76 winning line sets of Qubic. Particularly, the new GP is made to pull from the fourteen handpicked solutions in the memory list of Table 2.

Table 2: Lookup Table of Solution Sets.

First Set of Solutions	Second Set of Solutions
{18,32,46,60}	{0,26,52,78}
{3,27,51,75}	{15,31,47,63}
{5,26,47,68}	{6,26,46,66}
{22,27,32,37}	{7,27,47,67}
{18,33,48,63}	{17,32,47,62}
{20,21,22,23}	{25,26,27,28}
{65,66,67,68}	{70,71,72,73}

The modified version of the is_greater primitive function takes the first or second element from the input list l, depending on the greater of the values, and applies the modulus operator to regularize the number as a valid solution in the lookup table:

- is_greater(l): if $l_0 > l_1$: return l_0 mod 7 from S_1 else: return l_1 mod 7 from S_2

The first two elements in the first and second set of solutions $S1$, $S2$ were chosen for the reason these lines contain four rich points (seven winning lines running through them). Two of which are the richest points on the board (i.e., the centers of the middle boards). Players past the beginning skill level of Qubic know perfect players open their games by playing on rich points. The rest of the solutions in the list are randomly chosen. Comparable results would be generated using different sets of solutions. This solves the second problem. Lastly, to offset the domination of the heuristic player, the two-step evolution process is applied to solve the third main problem.

5.4 Two-Pass Evolution

Two-pass Evolution models the behavior of bucks (male deer) in the wild and may answer the third problem. Bucks compete by colliding antlers with other bucks where the act of competition decides dominance, and the winning males mate [9]. In the artificial model, colliding antlers is naturally equivalent to competing programs to decide dominance, and this is accomplished with first-pass evolution. The dominant program is now ready and allowed to mate. In the artificial world, mating is equivalent to throwing the best player, along with the heuristic player (predator), into the initial set of programs within second-pass evolution. In this second pass of evolution, the best GP player and predator mate. Hopefully, the mating produces better offspring through genetic evolution by applying genetic operators, recombination and mutation, to the best GP player with good players from the previous generation. By including the heuristic player as one of the players in the initial population of programs, the GP will eventually learn to block from the bad experience of losing to the predator, if reinforced with a win. This approach causes the evolving player to learn from others outside of its own class. Once the GP has evolved with the predator in the population, it is ready to compete in the real world. The aim is to evolve the GP into a stronger player capable to win more than a couple of games out of another 100 game contest of Qubic.

6 Next Experiment

6.1 Experiment Results: Second Attempt

The GP won 5 out of another 100 game contest of Qubic. Earlier, it was found, the GP was unable to solve the problem on its own. Now that the human programmer provided the right basic functions, modification of one of the primitive functions, and the addition of the equal blocking logic, it worked. The GP player defeated the heuristic player. The GP was able to find the right combination to pull the moves in by discovering the best way to arrange and the best order to call the solution sets. The most evolved player to date including second-pass evolution and selective termination achieved 25 wins and 12 draws, slowly approaching equality with the heuristic player. As a result of adding defense and second-pass evolution, the players start to become equal and longer games emerge.

6.2 Corrections to the Plan

Once the individual games start becoming long, the GP cannot find a valid move from the solutions (answers) list, all moves have already been taken, and the GP ends up timing out, or rather over-evaluating, and must give away the game to the opponent. There are only a maximum of 28 moves to choose from the list. If the opponent takes some of the moves away then there are less. Adding more solution lists to the specialized primitive function is not the answer. The set of solutions needs to be small enough to make sequential moves from within the same lists. Creating seven solution lists gives "optimal" success. For this reason, the GP had to generate random moves to complete close end games. The random move generator starts generating random moves until the move is allowed. For instance, if GP was the X player, it was X's move, the game was already 60 ply deep, and the GP's list of solutions were exhausted on the 61st ply, then the random move generator would kick in and return random moves for the GP to successfully complete the game.

7 Future Work

The evaluation function, as well as it plays, has some flaws, and can be improved upon. First, the heuristic player is being overcautious, playing unnecessary defensive moves, before spotting a winning offense. Small corrections in the evaluation function will change the function's behavior. Greater improvements can be made in the area of inserting further expert knowledge into the evaluation function. For instance, it is safe to say that making three blindfolded corner moves on rich points at the beginning of the game is a strong play. Another possible improvement could be to implement the concept of collecting threats, where the number of threats is more important than the threat itself [6]. Significant improvements can be made to the GP. For instance, the list of solutions referenced by the primitive function can be rearranged. Moreover, an improved method could be implemented for the times when the GP runs out of solutions in the end game. After these changes are made, the challenge would of course be to further develop the Qubic GP to dominate the heuristic player. After this, the next challenger to try and beat would be Lutz Tautenhahn's Qubic AI at http://www.lutanho.net/stroke/play.html. It is beatable on expert level, although, it plays extremely well and plays stronger than the Open-123 heuristic player.

8 Final Conclusions

The original hypothesis of developing a strong GP player for competing in the Qubic competition failed in some respects. One major problem was discovered in the course of the experiment. The GP player lost every game. The corrections to the GP player resulted in more than a few defeats of the heuristic player. An intelligent Qubic player was developed to beat the heuristic in 25 out of 100 games of Qubic. This was achieved without adding considerable expert knowledge to the GP. The GP plays at a higher skill level than its author; however, the end result of the contest, in terms of the number of wins, finds the heuristic player as the better Qubic player. It was found, the human programmer needs to supply basic needs, the right initial set of functions, and the right environment for the GP to evolve.

9 References

[1] B. Challenor. “Qubic2: 4x4x4 tictactoe, ti-83/84 plus assembly games (mirageos)”; [Online]. Available: http://www.ticalc.org/pub/83plus/asm/games/mirageos/
April 2012.

[2] J. Beck. “Combinatorial Games: Tic-Tac-Toe Theory”. Cambridge University Press, 2008.

[3] N. Do, “Mathellaneous: How to win at tic-tac-toe”; Gazette of the Australian Mathematical Society, 32, 3, 159, July 2005.

[4] O. Patashnick, “Qubic: 4 x 4 x 4 tic-tac-toe”; Mathematics Magazine, 33, 4, 151—161, Sept 1980.

[5] L. V. Allis, “Searching for Solutions in Games and Artificial Intelligence,” Phd dissertation, University of Limburg, 1994.

[6] G. Vitko, Oakland University Virtual Reality Lab, Rochester, MI, private communication, Feb 2009.

[7] J. Giarratano and G. Riley. “EXPERT SYSTEMS Principles and Programming”. PWS Publishing Company, 1998, 8.

[8] T. Segeran. “Programming Collective Intelligence. 1st ed.”. O’Reilly Media, Inc., 2007, 250–273.

[9] W. Embar, “Animal facts – deer”; [Online]. Available: http://www.veganpeace.com/animal_facts/Deer.htm
July 2005.