Heuri: A Scrabble Playing Engine Using a Probability-Based Heuristic

Alejandro González Romero [a,*], René Alquézar Mancho [a], Arturo Ramírez Flores [b], Francisco González Acuña [c,d] and Ian García Olmedo [e]

[a] *Ciències de la Computació, Universitat Politècnica de Catalunya, Barcelona, Espanya*
E-mails: yarnalito@gmail.com, alquezar@cs.upc.edu
[b] *Ciencias de la Computación, Centro de Investigación en Matemáticas A.C., Guanajuato, México*
E-mail: ramirez@cimat.mx
[c] *Matemáticas Básicas, Centro de Investigación en Matemáticas A.C., Guanajuato, Mexico*
E-mail: ficomx@yahoo.com.mx
[d] *Instituto de Matemáticas (Unidad Cuernavaca), Universidad Nacional Autónoma de México, Cuernavaca, Morelos, México*
[e] *Universidad Nacional Autónoma de México, Ciudad de México, México*
E-mail: ian.garcia@gmail.com

Abstract. The game of Scrabble has been successfully tackled by two engines: Quackle and Maven. They attain the state-of-the-art in Computer Scrabble. These engines use simulation techniques and precalculated values to achieve superhuman play. This paper presents a Scrabble-playing engine, Heuri, which achieves world championship standards when playing in Spanish, and a very high level when playing in English and French, without using brute-force approaches.One advantage of Heuri is that its calculations, as opposed to many in Quackle, are not precomputed and instead performed for every turn. Spanish match results of different versions of Heuri against humans and two versions of Quackle are presented in this paper. Results against Quackle, in English and French, are also given. The main goal of this paper is to enrich Computer Scrabble with the description of a Scrabble playing program, Heuri, which is differently designed from earlier ones such as Maven and Quackle. An additional goal is to give Scrabble players an excellent strategy for obtaining championship level in Spanish.

Keywords: Scrabble, bingo, heuristic, bingo probability, Heuri, leave

1. INTRODUCTION

Scrabble is a word game in which, in its tournament version, two players score points by placing tiles, each tile representing a letter to form words on a board, rather like in a crossword. World Championship Tournaments are played in many languages, such as English, French, and Spanish, English being the most common.

The AI game tradition focuses on games with perfect information, with Go, Chess, Checkers, Lines of Action, Reverse, Awari, Hex, Renju, Amazons, Pente, etc. dominating the literature.

A body of literature also exists on imperfect information games (in other words, those with private information), such as Scrabble, Poker and Bridge, to which this article contributes.

It is debatable whether a state-of-the-art Scrabble computer program beats the best human. In Wiles (2018) Howard Warner states: “Nigel Richards, the top player in the world, seems to have computer for a brain. In many ways we think he could actually be superior to Quackle the program.”

“Editorial note: According to FiveThirty Eight’s analysis of cross.tables.com’s data, Richards is technically ranked higher than Quackle.”(Wiles, 2018).

There is only one head-to-head encounter between Nigel Richards and Quackle, it took place in the Toronto International Scrabble Open 2011, Nigel won 473-449.

Spanish Scrabble players are not as good as English Scrabble players, perhaps because English Scrabble has been around since the mid 1970’s, and Spanish Scrabble since around 2000. Spanish Scrabble players can learn a lot from playing and watching games using strong bots. We think that humans can learn a lot from a Scrabble Heuristic bot such as Heuri.

Even though Heuri can play in any language, if a dictionary is plugged into it, it performs better with highly inflected languages, such as Spanish and French, and is less effective in languages whose verbs have few inflections such as English. The goal of this article is to enrich Computer Scrabble with the description of a Scrabble playing program, Heuri, designed differently from previous ones such as Maven and Quackle. An additional goal of this paper is to give Scrabble players an excellent strategy for achieving championship level in Spanish.

Heuri is a Type B program. According to Shannon (1950), a Type B program is a selective approach to searching in minimax trees which only considers a subset of plausible moves, in contrast to brute-force, Type A programs. Quackle, which has a state-of-the-art stature in Computer Scrabble, has an engine that is closer to a Type A program. Quackle plays hundreds of thousands of self-play games to pre-calculate the values of Superleaves, which are evaluations of all strings of length 6 or less. On the other hand, Heuri calculates the values of strings, of length 6 or less, the moment a player has to make a move (with the actual board).

Heuri has achieved extremely good results in Spanish, against humans and Quackle, and reasonable results in English or French, for a Type B program, against Quackle.

The article is organized into several sections. Section 2 mentions the move generator methods, describes the state-of-the-art in Computer Scrabble, and includes some results of Heuri matches in Spanish against top-level Scrabble players and Quackle. Section 3 refers to the first Spanish Scrabble program played with the official Federación Internacional de Scrabble en Español (FISE) rules. Section 4 deals with the heuristics used in the first, second and third versions of Heuri and includes evaluation formulas as well as the fishing technique. Match results are provided in Section 5, together with a comparison of results using different languages. Lastly, Section 6 contains conclusions and suggestions for future research.

2. SCRABBLE ENGINES

In the 1980’s when computers had less memory, Scrabble engines used a Direct Acyclic Word Graph, DAWG (Appel and Jacobson,1988), an efficient data structure for storing the Scrabble lexicon, which was both ingenious and memory saving. Today’s computer memories, however, make it possible to store Scrabble lexicons in several files with far more memory than DAWGs. The Anagram Method (González et al., 2018) uses this type of storage to increase the speed of move generation. With the Anagram Method, move generation becomes faster than traditional move generator methods using DAWG or GADDAG (Directed Acyclic Graph prefixed by its own inverse) see Gordon (1994).

Once a move generator exists, the implementation of basic engines based on the move that maximizes the score in each turn is the next logical step. Shapiro (1979) and Stuart (1982) are examples of forerunners in the development of Scrabble engines. Simulation techniques were used to overcome the obvious shortcomings of this greedy approach. These techniques considered the possible replies to a player's candidate move, the replies to those replies, and so on for many plies. This method, known as simulation, relies on Monte Carlo sampling (Brownlee Jason, 2019) to generate the opponent's rack, in situations with uncertainty, its theoretical basis being the Minimax technique (Russell and Norvig, 2009) of Game Theory.

The Maven program (Sheppard, 2002) developed by Brian Sheppard is one of the references for this paradigm, as borne out by its excellent results against top-level players. Some years later, Katz-Brown and O'Laughlin (2007) implemented Quackle, a powerful artificial intelligence Scrabble player, which also exploits the simulation technique.

Maven constructs a data set, giving a numerical value to sets of tiles, typically of length 1,2,3 or 4. The evaluation of a leave is obtained, roughly, by merging this with other numbers involving heuristics, such as the quotient obtained by dividing the number of vowels by the number of consonants (Sheppard, 2002).

Quackle does the same thing conceptually, the difference being that it now considers numerical values for all strings of length less than 7. These values are calculated by playing hundreds of thousands of Quackle vs. Quackle games (Katz-Brown and O'Laughlin, 2007).

In addition to the leave evaluations mentioned above, the strong features of Maven and Quackle include their simulation procedures and, in the case of Quackle, the power of their Superleaves.

A weakness of Maven and Quackle is the fact that their calculations are precomputed, and do not take into account the actual board, as opposed to Heuri's calculations, which are performed every turn, using the actual board.

The authors have developed Heuri (González et al., 2012; Ramírez et al., 2009; González et al., 2008), an engine that uses the Anagram Method (González et al., 2018) to generate all the legal moves. Unlike Maven and Quackle, to date, Heuri has not used simulation to decide which move to make. Instead, Heuri uses a probability-based heuristic to choose the move to be played. Several matches against top-level Scrabble players have confirmed Heuri's high playing strength.

Heuri's most important match victory against humans, was against the 2006 and 2008 World Scrabble Champion Enric Hernandez. Heuri defeated the champion 6-0!

In Spanish, Heuri played a 2000-game match against Quackle using the Speedy Player, winning 61% of the games. The Speedy Player evaluates potential moves without forward looking, and Quackle is weaker in Spanish since there are no pre-computed leave values (estimated values of the letters left on the player's rack) for Spanish. Leave values are a key feature of Quackle's playing strength.

Heuri also played a 1014-game match against Quackle using the 20-Second Championship Player bot. This is a much stronger bot than the Speedy Player one, since it uses a 20-second simulation for every move. Heuri won 53.35% of the games.

3. THE BEGINNING

3.1. A Spanish Scrabble Program

After a great deal of work, the enormous task of building the lexicon was completed. We then modified Scrabbler, a free English Scrabble source program built by Amitabh in 1999. The double letters CH, LL, RR were added as well as the letter Ñ. The distribution was also changed and finally some modifications had to be made to adapt to the rules for Scrabble tournaments in Spanish, involving the ability of a player to choose whether to change a certain number of tiles or to pass.

Eventually, we had the first Spanish Scrabble program ready to play with the official Spanish Scrabble tournament rules (FISE rules). Let us call this program Scrabbler II.

To check our Spanish lexicon and the functionality of the Scrabbler II program, a match was played between Scrabbler II and an expert human Scrabble player ranked 28th in the Spanish Scrabble league and 90th in World Spanish Scrabble. The match followed the format of a baseball World Series match. In other words, the first player to win four games wins the match. The human expert won 4-2.

Although at first sight one might think that a computer has an enormous advantage over a human because it knows all the permissible words, it turns out that this does not suffice to beat a Scrabble expert. This is because Scrabbler II adopts a naive, greedy strategy. It always plays a move that will score the highest number of points at that moment. This is an optimal strategy for duplicate Scrabble, where the aim of the game is to make a move that obtains the most points every turn. However, this is not a good strategy for normal Scrabble, since it ignores how this could affect the possibility of making better moves in future plays. Scrabbler II often ends up with poor quality racks such as GGLHQPM and since there is no exchange strategy, these racks will tend to prevail, giving the human player an enormous advantage. For instance, in Scrabble, a good player usually tries to play a bingo (all seven tiles in one turn), since this gives 50 bonus points. It is therefore advisable to keep the blank tiles, which can be played representing a single chosen permissible letter. Since Scrabbler II always plays a move with the highest possible score, it tends to get rid of blank tiles without necessarily making a bingo.

Since all these features were observed, we decided to design an improved version of Scrabbler II with an inference strategy engine. This engine uses a heuristic function that attempts to make high scoring moves while balancing the rack.

4. MOVE EVALUATOR

4.1. Introduction to Heuri

The first step towards building a Scrabble engine, with an inference strategy that could balance the rack in play, was a C function that received a 7-letter string and calculated the probability of making a bingo, for all 128 possible exchanges of the given string. To this end, we built a file, the septets, containing all the 7-letter strings that produce a bingo. An example of how to calculate this probability is given in section 4.2.

An essential part of evaluating a particular move involves assessing what will be left on the rack after the move has been made (the rack residue or leave). Since there is a 50-point bonus when you use all your tiles, a natural way to assess a move is to add its score to the expected value that the leave will be able to produce a bingo in the following move.

Several ideas about the heuristics used by humans when playing Scrabble, such as the use of anagrams, led to the construction of the computer lexicon and the Heuri engine.

4.2. Probability and heuristic of Heuri's first engine

An important part of a program that plays Scrabble in Spanish (or any other language) is deciding what to leave on the rack. In González et al. (2008), we provide a numerical evaluation as follows:

$$v = j + p * b - d \quad (1)$$

where j is the number of points made by the move, in which t tiles are played (it is assumed here that the bag has at least t tiles); $j = 0$ if the t tiles are exchanged rather than played on the board; p is the probability of obtaining a 7-letter bingo if t tiles are drawn at random from a set that is the union of the bag and the opponent's rack (which we call the "augmented bag"); b, the expected value of a bingo, is 77 (an average obtained from 1000 games) or rather:

$$b = 50 + 2.5(r + 1.92\ t) \quad (2)$$

where t is the number of tiles drawn from the bag and r is the total number of points in the leave; d is a nonnegative number, which is zero if the move is not weak from a defensive point of view. It is 20, say, if one places a tile between two triple word score squares, permitting a possible nine-timer, 10 if one plays a vowel next to a premium square allowing a six-timer, and 5 if the play permits a possible four-timer.

The equality for b was obtained as follows. First, let $b = 50 + k * e$ where e is the expected total of points (without multipliers or bonuses) of a bingo. The sum of the values of the 100 tiles is 192, meaning that the average value of a tile is 1.92 and $e = r + 1.92\ t$. For example, if the leave is {ZADO}, with $r = 14$, and we draw $t = 3$ tiles from the bag, we would expect these tiles to add, on average, $t * 1.92 = 3 * 1.92$. Say these tiles are {VES} (adding 6 points). Together with {ZADO} these tiles form VEZADOS, giving $e = 20$ (approximately $r + 1.92t$). But most bingos include premium squares (very often, in addition to the 50-point bonus, the word score is multiplied by 2 and sometimes 3, 4 or 9). It is therefore reasonable to multiply e by a constant k and we took $k = 2.5$ because it often yields a b value close to the experimental bingo average of 77.

For example, the frequent case where $r = 3$, $t = 4$ yields $b = 50 + 2.5(3 + 1.92 * 4) = 76.7$.

It is better to explain the calculation of p using an example:

What is the probability p_s of obtaining the "septet" s ={AAAAÑRR} (from which one can form the 7-letter bingo ARAÑARA) if one has a leave {AAAÑ}, exchanges {HQV}, the augmented bag is "total bag – {AAAÑHQV}" and draws three tiles from it?

Answer: If {AAAÑ} were not contained in {AAAAÑRR} p would be 0. However {AAAÑ} is contained in {AAAAÑRR} so we consider the different set {AAAAÑRR}-{AAAÑ}={ARR}={AR2} and the augmented bag: {AAAAAAAAARRRBBCCCC...XYZ}={A9R3B2C4...XYZ}. One then computes $p_s = C(9, 1) * C(3, 2) / C(93, 3)$ (93 is the number of tiles in the augmented bag and the 3 in $C(93, 3)$ is the number of tiles drawn from the bag) where $C(m, n)$ is the binomial coefficient:

$$C\ (m, n) = m\ (m - 1)\ (m - 2)\ \ldots\ (m - n + 1) / n! \quad (3)$$

The numerator has n factors and $C(m,n)$ is the number of n-element subsets of a set consisting of m elements. Note that the denominator $C(93,3)$ does not depend on the septet s.

The probability p of obtaining a 7-letter bingo if one has the leave {AAAÑ} and the augmented bag "total bag - {AAAÑHQV}" is therefore the sum of all p_s when s runs over all septets.

4.3. Heuri's second engine

Although Heuri's first engine has the essential part of the algorithm used by Heuri to evaluate a move, it required certain improvements. In addition to making the engine play better, these improvements enabled the engine to start playing automatically.

In Ramírez et al. (2009), it was decided to give a numerical evaluation of all potential moves as follows:

Let us recall some essential ideas from the last section and introduce some new features.

An important part of a program that plays Scrabble is the decision of what to leave on the rack. In a move t tiles are played or exchanged, and n - t tiles make up the leave r (excluding the endgame, $n = 7$).

The following heuristic function is used to evaluate all potential moves:

$$v = j + e - d \tag{4}$$

where j is the number of points made by the move, in which t tiles are played (it is assumed here that the bag has at least t tiles; $j = 0$ if t tiles are exchanged rather than played); e is the expected value of a bingo, given a leave r, if t tiles are drawn randomly from the augmented bag, in other words, the union of the bag and the opponent's rack; d is a nonnegative number, which is zero if the move is not weak from a defensive point of view. In (4), $d = 0$. Out of all the potential moves, one with maximal v is chosen.

To explain our estimate of e, a septet is defined as a lexicographically ordered string of seven characters of {A,B,...,Z,#} (where # is a blank) from which a 7-letter word can be constructed. For example, {AAAAÑRR} (yielding ARAÑARA) and {ACEINR#} (yielding RECIBAN) are septets but {AEEQRY#} and {ADEILOS} are not.

There are 126,972 septets. The following formula was used to calculate e:

$$e = \sum_{i=1}^{126,972} p_i (50 + k\,\sigma_i) \tag{5}$$

Let us call Heuri_2 the heuristic in (5). Heuri_2 is the heuristic used in the Heuri program presented in Ramírez et al., (2009).

Here, p_i is the probability (which could be zero) of obtaining the i-th septet, given a leave r consisting of 7 - t tiles, if t tiles are randomly drawn from the augmented bag; σ_i is the sum of the values of the characters in the i-th septet. The existence of premium squares, hook words, bingos of length greater than 7 and experimentation have led us to give 2.5 as the value of the constant k.

The calculation of p_i was explained (with an example) in section 4.2.

4.4. Heuri's third engine

Although the formula (5) is a good estimate of *e* when the board is open, it ignores the fact that one may have a septet on the rack which cannot be placed as a bingo on the board. This often happens when the board is closed. To account for this, one can write:

$$e = \sum_{i=1}^{126,972} \delta_i \, p_i (50 + k \, \sigma_i) \tag{6}$$

where δ_i, which depends on the board, is 1 if the *i*-th septet can be played as a bingo on the board and 0 otherwise; (the board is taken before the next opponent's move). The calculation of δ_i, for all *i*, which is independent of the rack, is feasible because Heuri has a fast move generator based on anagrams (González et al., 2018).

A collection of 1000 Scrabble games was amassed to classify the bingos played in those games into three categories by length: 1) seven-letter bingos, 2) eight-letter bingos, and 3) X-letter bingos, with X > 8. This classification yielded the following results:

The total number of bingos in the collection of 1000 games was 6073 bingos, divided as follows: 2003 seven-letter bingos, 3972 eight-letter bingos and 98 bingos with more than 8 letters. This reinforced a previous thought: the heuristic function should include the probability of making 8-letter as well as 7-letter bingos.

Let us define an *octet* as a lexicographically arranged string of eight characters of {A, B, ..., Z,#} (where # is a blank) from which an 8-letter word can be constructed.

A *reduced octet* is a lexicographically ordered string of seven characters of {A, B, ..., Z, #}, which forms an octet when adding a suitable new character. Let a *pure septet* be a septet that is not a reduced octet.

There are 126,972 septets, 5,830 pure septets and 282,214 reduced octets. Note that the set of septets contains all the pure septets and 121,142 reduced octets that are also septets. We wish to provide an estimate of the probability of obtaining a 7-or 8-letter bingo, given a leave and a board. The pure septets and the reduced octets yield a total of 288,044.

Then the calculation of *e* is given in González et al. (2012) by the formula:

$$e = \sum_{i=1}^{288,044} \delta_i \, p_i (50 + k \, \sigma_i) \tag{7}$$

Let Heuri_3 be the heuristic in (7); p_i, σ_i, k and δ_i were explained earlier.

4.5. Fishing

Fishing plays are important because they seek high scoring moves in subsequent turns. Brian Sheppard, the developer of Maven, writes the following in Sheppard (2002): "Maven's lack of a move generator for fishing may be its biggest weakness. All of the moves that Maven overlooked in its match against Adam Logan were fish." Heuri fishes extremely well, which is a key feature of a good Scrabble engine.

Intuitively, a fish is a move that seeks a bingo in the next turn. More precisely, using Heuri's probability heuristic, $v = j + e - d$, let us define a *fish* as a move where $e > j$ (its expected value of a bingo is greater than the points made by the move). The most common example of a fish is an exchange with $e > 0$. Almost all exchanges are fishing moves. However, in the pre-endgame (when there are only a few tiles in the bag), a player may change a Q tile to avoid being left with it at the end. It is likely that $e = 0$ since no more bingos can be placed on the board, or it is impossible to construct a bingo with the player's rack leave and the tiles left inside the bag (González et al., 2012).

The following is an example of a possible line of play, using Heuri, which shows and involves fishing moves. Figure 1 shows the fishing moves; the player's rack is retuuqh. The first six moves are fishing moves since $e > j$. All moves except move 7 are fishing moves since $e > j$ for all of them. In 7 hurte, $j = 24$ and $e = 9.515$ so $j > e$. (In figure 1, *p* represents the value of *e*).

Fig. 1. Fishing part 1.

After playing the best fishing move, (7G he) a letter "e" is drawn, from the bag, and the board position is shown in figure 2.

Fig. 2. Fishing part 2.

The next player then exchanges tiles. The player in turn has the following rack: retuuqe and the best move is shown in figure 3 (8A turquees which gives 106 points). Another equally good move is 8A truquees.

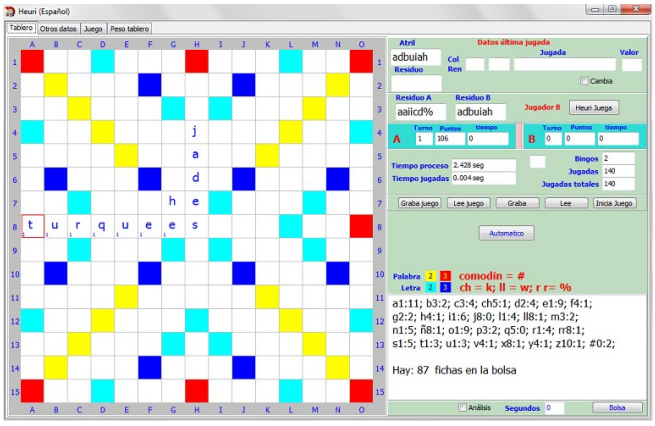

Fig. 3. Fishing part 3.

5. MATCH RESULTS

5.1. Match Results of Heuri's First Engine

Due to the unfinished state of the engine, we could only perform a few games to test our first heuristic. The first match Heuri played was a match against an engine that always played the highest move: Heuri won 4-0. Matches against human beings were performed on the Internet, which gave them an advantage since they could consult words before placing them on the board and also had a lot of time to play. In its first match against a human being, Heuri won 4-2 against one of the best players in Colombia, Fernando Manríquez, with an ELO of 2110 (the highest ELO in the FISE list was 2191), ranked No.20 in the FISE list of Scrabble players. Another important victory Heuri scored was against Benjamín Olaizola, from Venezuela, the reigning (2007) world champion who had also won the world championship in 2001. With an ELO of 2169, he ranked second in the international list. It was a very close game: (Heuri 471 Benjamín Olaizola 465). Eventually, Heuri lost the match to Olaizola 2-4. After Heuri played 17 games against several top-ranked players, the results were as follows: Heuri won 10 and lost 7, with an average of 509 points per game.

5.2. Match Results of Heuri's Second Engine

After playing 40 games against several top-ranked players, Heuri achieved the following results using the heuristic Heuri_2 described in section 4.3: 27 wins and 13 losses, with an average game score of 511 points. Almost all the matches followed the format of a baseball world series: the first opponent to score four victories won the match.

The most important match victory for Heuri was against 2008 World Champion Enric Hernández, from Spain, also World Champion in 2006. This was a longer match, consisting of a maximum of eleven games to decide the winner. Heuri defeated the World Champion 6-0! An excellent result. Heuri also won 4-1 a match against Airan Pérez from Venezuela, one of the best players in the world, and 2007 and 2008 Runner-up. Pérez has won two World Championships (in 2013 and 2015). Heuri also defeated Aglaia Constantin, then the best player of Colombia, twice National Champion of Colombia (2006 and 2009), 4-1. Heuri won four matches and lost one: the remainder of the games were incomplete matches against former world champions and national champions.

5.3. Match Results of Heuri's Third Engine

Using Quackle open-source code and Heuri, a connection was established to make it possible to play Spanish Scrabble games between the two engines. Since Quackle is not as strong in Spanish as it is in English, future matches will be played in English.

We will now present the results (see Table 1) of a 2000-game match in Spanish, played between Quackle using Speedy Player and Heuri's third engine. Speedy Player evaluates potential moves without any active forward looking, calculating only the short-term "utility" of a move: the value of the played word plus a pre-computed "leave" value, or the estimated value of the remaining tiles in combination with each other, plus a minor adjustment for the number and quality of locations now available to the opponent. Speedy Player is weaker in Spanish since there are no pre-computed leave values for Spanish.

Table 1
Heuri vs Quackle Matches in Spanish

Number of games	Heuri wins	Quackle wins
2000	1098	902

Heuri's third Engine also played a 507-game match against Quackle using 20 Sec. Champ Player (see Table 2). This is a much stronger bot, since it uses a 20-second simulation for every turn. The results show that Heuri won 53.35% of the games.

A *turnover* is a game where one player is losing when the endgame begins but wins after it finishes; (the endgame commences when there are no more tiles in the bag and ends when a player has no more tiles on the rack, or no move can be made). Table 2 shows that out of the 507 games in which Heuri began, 283 were won by Heuri and in 47 of these, Heuri was behind when the endgame started and ended up winning after the endgame had finished. Table 2 also shows that out of the 507 games in which Quackle started, 253 were won by Quackle and in 54 of these, Quackle was behind when the endgame started and ended up winning after it had finished.

Turnovers indicate an estimation of the quality of play in the endgame. The first player always has a significant advantage, and as can be seen from the table, Heuri plays better in Spanish than Quackle; nevertheless, Quackle plays endgames better.

Table 2
Heuri vs Quackle Matches in Spanish

1014 Games	Heuri	Quackle	Quackle	Heuri
Wins	283	224	253	254
Turnovers	47	12	54	5

5.4. Heuri plays better in Romance languages

Using the Heuri_3 heuristic, the Heuri Scrabble engine was set up so that matches between Heuri and Quackle could be played in French. The main reason for these experiments is to prove the following hypothesis: "Due to Heuri's bingo-oriented strategy, Heuri plays better in Romance languages such as Spanish and French than in English." Romance languages like Spanish and French have a high number of verb conjugations compared with

non-Romance languages like English. In other words, verb conjugation yields many words. For example, the verb *amar* produces nearly 50 different words in contrast to the verb *love*, which produces only four. Most 7- and 8-letter words are derived from verbs. The most common bingos are 7- and 8-letter words. Remember that a bingo is a move in Scrabble that uses all the tiles in a player's rack, and most importantly, provides a 50-point bonus, which is why playing bingos is crucial for winning a game. Due to the number of verb conjugations in Romance languages, games played in them tend to have more 7- and 8-letter words than those played in non-Romance languages. Moreover, most verbs in Romance languages contain frequent letters in Scrabble. Accordingly, 7- and 8-letter words that are verb conjugations are likely to appear in Scrabble games. For example, the most frequent bingos in Spanish are: OSEARIA, ASAETEO, ALETEADO derived from the verbs OSEAR, ASAETEAR and ALETEAR.

In Table 3, the results of the matches between Heuri and Quackle using the greedy approach are given. As can be seen from the following two tables (Tables 3 and 4), Heuri plays better in French and Spanish than English.

For the performance of Heuri vs. Greedy Opponent (that is, Quackle using a Greedy variant) after playing 10,000 games in each of the three different languages (English, French and Spanish), see Table 3.

Table 3
Matches Heuri vs Quackle Greedy

Heuri vs Quackle greedy	English	French	Spanish
Heuri winning percentage	62.17%	65.32%	64.7%

Let us recall certain characteristics of the Quackle Speedy engine. Quackle Speedy Player is a Scrabble engine that does not use simulation, although it does use Quackle's Superleaves when playing in English and French. Quackle's Superleaves are numerical values corresponding to different multisets of tiles (with a cardinality between 1 and 6). These Superleaves are precomputed values, calculated by playing hundreds of thousands of Quackle vs Quackle games. They were precomputed in English and French, but there are no Superleave values in Spanish.

Quackle Speedy Player outperforms Heuri in English and French due to the Superleaves, but Heuri wins in Spanish against Quackle Speedy Player because of the lack of Superleaves in Spanish; see Table 4.

Performance of Heuri vs. Quackle Speedy Player after playing 10,000 games:

Table 4
Matches Heuri vs Quackle Speedy

Heuri vs Quackle greedy	English	French	Spanish
Heuri winning percentage	45.52%	47.12%	63.83%

6. CONCLUSIONS AND FUTURE WORK

A novel Scrabble move generator based on anagrams has been designed and implemented, which is faster than the GADDAG-based generator used in Quackle engine.

A novel Computer Scrabble engine called Heuri has been designed and implemented, it uses specific heuristics not previously used in the state-of-the-art.

Heuristic evaluation functions of increasing playing performance, all of them based on probabilities, have been proposed and tested, which allow Heuri to beat Spanish World Champions Scrabble players, as well as (top-level) expert human players without the need of simulation techniques.

Heuri was able to beat the strong engine Quackle, in Spanish, even when Quackle used a 20-second simulation bot.

The heuristics mainly consider the possibility of achieving a bingo on the actual board, whereas Quackle used pre-calculated values (Superleaves) regardless of the latter. Heuri is primarily designed for Spanish, although it can play in English and French.

The heuristics used in Heuri proved effective as seen from their excellent results against human Scrabble Champions. There is still room for improvement in the heuristics, to obtain better results, in English and French, against the Quackle engine. For instance, the heuristics could include contemplating non-bingo high scoring moves.

The probability heuristic (7) can be improved by adjusting the constant value k (currently k =2.5). In the opening move, a few experiments indicated that k=2.7, but more experiments are still required. Moreover, k is unlikely to have a constant value since it probably changes throughout the game.

Simulating one or two moves ahead (1 or 2 plies) would be one way to improve results.

Using only one move ahead (1 ply) would consider a defense value, thereby obtaining a value for $d > 0$, rather than simply taking $d = 0$ (see formula $v = j + e - d$ (4) of subsection 4.3).

Going even further by looking two moves ahead (2-plies), simulating one move by the opponent and our response, would probably achieve better results than just by looking one move ahead, while striking a good balance between defending and attacking.

Lastly, while analyzing games, it is possible to seek common situations and attempt to build a defense for them. An important feature is to incorporate the opponent's last move to attempt to deduce information about the opponent's rack. In other words, it is possible to include opponent modeling in the pre-calculations of the defense and record key defensive strategies while analyzing games.

REFERENCES

Appel, A.W., Jacobson, G.J. (1988). The World's Fastest Scrabble Program. *Communications of the ACM*, 31(5), 572-578.

Brownlee, Jason. (Nov., 2019). A Gentle Introduction to Monte Carlo Sampling for Probability. https://machinelearningmastery.com/monte-carlo-sampling-for-probability/

González Romero Alejandro, González Acuña Francisco, Ramírez Flores Arturo, Roldán Aguilar Amador, Alquézar Mancho René and Hernández Enric. A Heuristic for Scrabble Based in Probability. (July 21-25, 2008). *Proceedings of the Workshop on Artifical Intelligence in Games AIG-2008. 18th European Conference on A.I. ECAI 2008, Patras, Greece*, 35-39.

González Romero Alejandro, Alquézar Mancho René, Ramírez Flores Arturo, González Acuña Francisco. (August 27-31, 2012). Heuristics and Fishing in Scrabble. *European Conference on Artificial Intelligence ECAI 2012. Proceedings of the Computer Games Workshop 2012 (CGW 2012). Montpellier, France*, 62-70.

González Romero Alejandro, Alquézar Mancho René, Ramírez Flores Arturo, González Acuña Francisco, García Olmedo Ian. (December, 2018). El método de anagramas: un rápido y novedoso algoritmo para generar jugadas de scrabble. *Research in Computing Science* 147(6), 41-55.

Gordon-Steven, A.A. (1994). A Faster Scrabble Move Generation Algorithm. *SoftwarePractice and Experience*, 24(2), 219-232.

Katz Brown J. and O'Laughlin J., (2007). "How Quackle Plays Scrabble". [Online]. Available: http://people.csail.mit.edu/jasonkb/quackle/doc/how_quackle_plays_scrabble.html

Ramírez Flores Arturo, González Acuña Francisco, González Romero Alejandro, René Alquézar Mancho, Hernández Enric, Roldán Aguilar Amador and García Olmedo Ian (November, 2009). A Scrabble Heuristic Based on Probability That Performs at Championship Level. *MICAI 2009: Advances in Artificial Intelligence. Lecture Notes in Artificial Intelligence 5845. 8th Mexican Conference on AI. Proceedings. Guanajuato, Mexico.* 112-123.

Russell Stuart J., Norvig Peter (2009). Artificial Intelligence: A Modern Approach. *Prentice Hall*.

Shannon, Claude. (1950) Programming a computer for playing chess. *Philosophical Magazine, Ser.* 7, 41, 256-275.

Shapiro, S.C. (1979). A scrabble crossword game playing program. *Proceedings of the Sixth IJCAI,* 797-799.

Sheppard Brian, Towards Perfect Play of Scrabble (July 2002). PhD thesis, *IKAT/Computer Science Department, Universiteit Maastricht*.

Stuart, S.C. (April 1982). Scrabble crossword game playing programs. *SIGArt Newsletter*, 80. 109-110.

Wiles Jamie. (March, 2018). Tilin'Out: A Conversation With A Champion Scrabble Player. https://www.urbo.com/content/tilin-out-a-conversation-with-a-champion-scrabble-player/.