Cheap-Talk Equilibria in Multidimensional Signaling Games with Non-Uniformly Distributed Types: Simulations and Applications

Candidate: Ružica RAKIĆ

Supervisor: Prof. Dr. Frank RIEDEL

July 31, 2010

Matriculation number: 1950597
E-mail: ruzica.rakic@gmail.com
# Contents

1 Introduction ................................. 1

2 Signaling games ............................. 5  
   2.1 Basic Model ............................... 5  
   2.2 Perfect Bayesian Equilibrium ............... 6  
   2.3 Cheap talk games .......................... 8  
   2.4 Example: Simple language game .......... 8

3 Voronoi Languages .......................... 11  
   3.1 The Model ................................. 11  
   3.2 Cheap-Talk Equilibria ...................... 11  
   3.3 Nash Equilibria ............................ 16

4 Evolutionary Game Theory .................. 19  
   4.1 Evolutionary dynamics ..................... 19  
      4.1.1 Discrete strategy space ............... 19  
      4.1.2 Continuous strategy space ............ 22  
   4.2 Static evolutionary concepts ............... 23  
   4.3 Stability of Voronoi languages .......... 24

5 The Algorithm and Simulations .......... 27  
   5.1 The Algorithm .............................. 27  
   5.2 Simulations - Different types of distributions 29  
   5.3 Applications ............................... 32

6 Zipf’s law ................................... 35

7 Conclusions .................................. 40

A Appendix ..................................... 41  
   A.1 Code for the Voronoi languages .......... 41  
   A.2 Code for the Zipf’s law .................. 50

Bibliography ................................ 52
Chapter 1

Introduction

In the everyday communication it is very common that people convey or can observe much more information than they are able to say. In the signaling games, however, it is usually assumed that there is the same amount of information as there are available words. In that case it is possible to attach each type to one word and get a perfect understanding.

In cases when the number of types is much bigger than the number of available words, the sender is obliged to say the same word for many different types that he conveys. One can ask then, what is the best way to attach different types with some word, or how to make some partition of the type set and each cell of the partition assign to a different word?

As an illustration, the first panel of Figure 1.1 shows an example with a color panel where every pixel has different RGB representation (intensity of red, green and blue color). The sender’s type can be any of these pixels, but the problem is that the receiver can understand only 8 colors (yellow, orange, red, pink, purple, blue, turquoise and green). Therefore, the sender has to make some optimal partition of the color panel and for many different colors to say only one word. One of the possible partitions is presented in the second panel of Figure 1.1. The partition is not unique, and our task is to find the best one.

Jäger, Metzger and Riedel in the paper "Voronoi Languages: Equilibria in Cheap-Talk Games with High-Dimensional Types and Few Signals" [Frank Riedel 2009] proposed a solution. They assumed that the type set is continuum, a convex subset of \( n \)-dimensional Euclidian space, and that there are only finitely many words. Given that this is a language game, players have the same interest, i.e. they aim to understand each other the best they can. They want to minimize the "misunderstanding" which is measured via some loss function that calculates the distance between the sender’s type
and the receiver’s interpretation. The authors have presented a solution that players can achieve in cooperation. The solution they found is called Voronoi language, because the sender is using Voronoi tessellation to make a partition of the type space. The receiver’s reply is the average type of each Voronoi cell, and is called the Bayesian estimator.

Furthermore, authors gave a full characterization of strict Nash equilibria of this game. Finally, given that the language is rather a dynamical structure, they analyzed the stability properties of the observed cheap-talk equilibria.

However, the analytical analysis of equilibria is not possible when the language contains more than three words, and they were obliged to rely on simulations. They proposed the algorithm that searches for efficient and stable languages from the big variety of languages with uniformly distributed type set.

This thesis gives the extension of their program applied to differently distributed types. Also, the program is written in a different programing

---

1Given a set of two or more but a finite number of distinct points in the Euclidian plane, we associate all locations in the space with the closest member(s) of the point set with respect to the Euclidian distance. The result is a tessellation of the plane into a set of regions associated with the members of the point set. We call this tessellation the planar ordinary Voronoi diagram generated by the point set (generators), and the regions constituting the Voronoi diagram ordinary Voronoi polygons (cells).
language (C++) which runs much faster that the one presented in the cited paper.

Given that we are here analyzing a language game, one can be interested if such languages satisfy some properties of the "live" languages. One of the most significant properties of "live" languages is Zipf’s law. That states that there exists a vocabulary balance in every corpora of natural languages. It states that the ratio between rank of each word from some corpus and its frequency is constant. In other words, the most frequent word from some corpus occurs twice as often than the second most frequent word, and so on.

It is obvious that Voronoi languages with uniformly distributed type sets do not follow this law, given that all the words in such a language are equally frequent. This is one of the reasons why we extended our program to work for differently distributed type sets. Our analysis suggests that arbitrarily chosen languages from exponential distribution are satisfying this property better that stable Voronoi languages with the same distribution. Other distributions gave even less satisfying results.

This paper is organized as follows. In Chapter 2 we introduce the signaling games based mostly on the approach of Fudenberg and Tirole [Fudenberg 1991] and and Gibbons [Gibbons 1992], and we give a definition of Perfect Bayesian Equilibrium, usually used for solving the signaling and other sequential games with non-complete information. Chapter 3 summaries the results from the paper by [Frank Riedel 2009] about the equilibrium that is achieved in the cooperation. Fundamental principles of the evolutionary game theory are given in Chapter 4. There are presented evolutionary dynamics and static evolutionary concepts. There is also given the stability analysis of the cheap-talk equilibrium derived in the previous chapter. The used literature is [Fudenberg 1998], [Gintis 2000], [Oechssler 2002], [Oechssler 2000], [Hofbauer 2003], [Hofbauer 2000], [Pawlowitsch 2008], and many others.

Chapter 5 contains the algorithm that selects efficient and stable languages among a variety of Voronoi languages. There are presented the results of the program which work for different distributions and also applications of the presented program. The applications are mostly found in [Qiang Du 1999].
Chapter 6 shows analysis of whether any of the Voronoi languages is satisfies the Zipf’s law, [Zipf 1972]. Chapter 7 presents the conclusions.


2.1 Basic Model

Signaling game is a game with non-symmetric incomplete information. In this game there are only two players, sender and receiver. Sender has its own private information about his type, and these are unknown to the receiver. Receiver only has a prior belief about the sender’s type, which is common knowledge. Given the type, sender chooses a signal which he sends to the receiver. The receiver, after observing the signal, updates his beliefs and makes an action.

Signaling game is a sequential game. This means that at first one player chooses an action, and after observing what the first player played, the second player chooses his actions. There are three steps in signaling game (timing of the game):

- The nature is choosing the sender’s type from the set of feasible types according to a probability distribution.

- The sender becomes aware of his type and chooses a signal from the set of feasible signals.

- After the receiver observes the sender’s action, but not the type, he updates his beliefs about the sender’s type according to the signal he got from the sender, and chooses an action from the action set.

Formally, signaling game in the normal form looks like, 
\[\{S, R\}, \{W, A\}, \{U_S, U_R\}, T, P\]. Here, \{S, R\} is the set of players, S stands for the sender and R for the receiver. \[W = \{w_1, ..., w_N\}\] is then a finite strategy set of the sender (set of all available words), and \[A = \{a_1, ..., a_M\}\] receiver’s strategy set (set of all interpretations). \[U_S, U_R : T \times W \times A \rightarrow \mathbb{R}\] are payoffs or utilities of the sender and receiver, respectively. \[T = \{t_1, ..., t_I\}\]
denotes the set of all possible types of the sender. \( p(t_i) \) is probability distribution of types over the type set, which is common knowledge, \( p(t_i) > 0, \forall i \) and \( \sum_{t_i \in T} p(t_i) = 1. \)

2.2 Perfect Bayesian Equilibrium

The equilibrium concept that is used for solving signaling games is perfect Bayesian equilibrium, which is one of the refinements of the Nash equilibrium. Main characteristics of this equilibria are sequential rationality and consistent beliefs. Sequential rationality signifies that in every moment of the game both players are trying to maximize their own payoffs for given beliefs. Consistent beliefs refer to updating of receiver’s prior beliefs about the sender’s type. So, receiver should modify his original beliefs conditional to sender’s action.

Receiver, after observing the signal \( w_n \in W \), has beliefs which types of sender could have sent message \( w_n \). Let us denote this belief by the probability distribution \( \mu(t_i|w_n) \), with \( \mu(t_i|w_n) \geq 0, \forall t_i \in T \), and \( \sum_{t_i \in T} \mu(t_i|w_n) = 1. \)

**Definition 1.** A (pure strategy) perfect Bayesian equilibrium in signaling game consists of \((w^*(t_i), a^*(w_n), \mu^*(t_i|w_n))\), where \( w^* \) is optimal sender’s strategy, \( a^* \) optimal receiver’s strategy and \( \mu^* \) optimal system of beliefs, and they are given with:

1. \( a^*(w_n) = \max_{a_m \in A} E[U_R(t_i, w_n, a_m)|\mu(t_i|w_n)] \)
2. \( w^*(t_i) = \max_{w_n \in W} U_S(t_i, w_n, a^*(w_n)) \)
3. \( \mu^*(t_i|w_n) = \frac{p(t_i)}{\sum_{t_i \in T} p(t_i)}. \)

The first requirement in the definition states that receiver upon observing some message \( w_n \in W \), maximizes his expected utility for given beliefs about the sender’s type. Second tells that the sender, for each type \( t_i \in T \), maximizes his utility, given that receiver plays optimally. The last requirement gives the optimal system of beliefs. For each \( w_n \in W \), if there exists \( t_i \in T \) such that \( w^*(t_i) = w_n \), then the receiver’s beliefs at the information
set corresponding to \( w_n \) must follow from Bayes’ rule and the sender’s strategy.

There are always plenty of equilibria in the signaling games and in the following we present the most general characterization.

**Separating equilibria**

Separating equilibrium is an equilibria in pure strategies. It can only occur if there are as many elements in the type set as signals in the signal set, \(#(T) = #(W)\). Each type of the sender chooses a different signal and then, after observing the sender’s action, the receiver knows the sender’s type. This leads to the perfect information transmission.

**Pooling equilibria**

In cases when the sender chooses always the same signal, irrespective of the type that he conveys, this leads to pooling equilibria. In this situation, sender’s signal gives no clue to the receiver about the sender’s type. There is no information transmission. There are also semipooling equilibria when the sender provides partial, but not full, information about his type. Evolution kills pooling equilibria.

**Babbling equilibria**

Babbling equilibria are completely mixed equilibria. If the sender is not reliable, thus says every word with some positive probability (so uses completely mixed strategies), this makes the receiver indifferent after hearing the message. Receiver then only relies on the prior beliefs, so the sender’s signal does not make any influence. This is called babbling a equilibrium. There is no information transmission in this equilibrium.

There are two different types of signaling games, the games with common interest, and with conflict of interest. In the first, players desire the same outcomes (for example, language games where both players want to understand each other as good as possible) and in the second, receiver does not want sender to reveal his type (for example, Spenser’s job market game).
2.3 Cheap talk games

Cheap talk games are almost the same as the signaling games, but the signals are costless and nonverifiable, and they do not have direct influence on payoffs of the game. The payoffs depend only on the receiver’s action and the prior beliefs. Truthfulness is not assumed. One could ask when is it reasonable to have the cheap talk? Cheap talk is informative only in the games with common interest, when both of the players are rational.

The timing of the cheap talk games is the same as in the signaling games, the only difference is that payoffs depend only on the receiver’s action and the type of the sender, so $U_S, U_R : T \times A \to \mathbb{R}$. So, the signals do not have direct effect on payoffs of both players. The only way in which the messages can influence the payoffs is through changing the receiver’s beliefs about the sender’s type. Because the cheap-talk games have the same timing as the signaling games, the definitions of perfect Bayesian equilibrium in the two games are also the same.

There are always babbling equilibria in cheap talk games. When sender uses mixed strategies, receiver only relies on prior beliefs, in which case the cheap talk does not matter.

2.4 Example: Simple language game

The game that I am analyzing in the following chapters is the language game of common interest. I will present here the simplest example of the language game, i.e. the game with only two types and two words.

Figure 2.1 gives graphical interpretation of this game with only pure strategies. Nature ($N$) is choosing the type, $t \in \{0, 1\}$, of the speaker ($S$), according to probability distribution, $(p, 1 - p)$. After the speaker learns his type, he chooses either to send signal $w1$ or the signal $w2$. After receiving the signal, receiver ($R$) updates his prior beliefs and interprets the signal as 0 or 1. $[b1]$ and $[b2]$ are beliefs of the receiver, i.e. they reflect his beliefs on the probability that the sender will play $w1$ or $w2$ respectively. Payoffs are: 1 when the type is equal with interpretation, and 0 otherwise. Figure 2.1 shows only
There are two separating equilibria of this game. The first is when sender says $w_1$ if he is of the type 0, and $w_2$ if he is of the type 1. In this case, both of receiver’s beliefs are determined by Bayes rule and sender’s strategy: $b_1 = 1$ and $b_2 = 0$. Receiver’s best reply on $w_1$ is 0, and on $w_2$ is 1. The payoffs for both players are 1. Deviations would lead to less payoffs, thus this is separating perfect Bayesian equilibrium. Another separating equilibrium is when $w_1$ and $w_2$ interchange.

Let us now analyze the pooling equilibria. Suppose that sender always says $w_1$ and that $p > 1/2$. Receiver’s beliefs are determined by Bayes rule and sender’s strategy, so $b_1 = p$. Given beliefs, receiver interprets signal as 0, and payoffs as 1. Nevertheless, we have to specify other possible outcomes of the game:

- if the sender is of type 1 and plays $w_2$, the payoffs are 0, but if he plays $w_1$ the payoffs are again 0. So, in this case it does not pay off for the sender to deviate;
- if the sender is of type 0 and plays $w_2$, the payoffs are 0.

We see that no improvement is possible. The receiver always understands 1, so the language is useless. Nevertheless, $(w_1, 0)$ is equilibrium, but pooling
perfect Bayesian equilibrium. Another pooling equilibrium is when sender always says $w_2$.

The only situation that remains is when sender wants to make receiver indifferent after hearing a signal, say $w_1$, then receiver’s beliefs are $b_1 = 1/2$. From Bayesian rationality, $b_1 = \frac{p \cdot k_1}{p \cdot k_1 + (1 - p) \cdot k_2} = \frac{1}{2}$, follows $p \cdot k_1 = (1 - p) \cdot k_2$. On the other hand, if sender says $w_2$, then $b_2 = 1/2$ and $p \cdot (1 - k_1) = (1 - p)(1 - k_2)$. By summing up the last two equations, we get $1 - p = p$, so $p = 1/2$. There exists babbling equilibrium only for $p = 1/2$. 

Figure 2.2: Mixed strategies in the language game
Chapter 3
Voronoi Languages

In this section, we present a model of an observed game, and derivation of cheap talk equilibria that are presented in the paper [Frank Riedel 2009].

3.1 The Model

Game that is analyzed here is a language game. The idea to use models of game theory to analyze occurrence of language, was introduced very early in the history of game theory by Lewis (1969). However the game that is analyzed here is more complicated, given that the type set is continuum, subset of multidimensional Euclidian space. Other settings are the same as in the approach given in the previous chapter.

In the normal form, this game may be seen as $[\{S, R\}, (\Sigma, T^N), U, T, F]$, where $\Sigma$ is the set of all (pure) senders strategies $w : T \to W$, and $W = (w_1, ..., w_N)$ is the finite set of signals. A (pure) strategy for the receiver is a vector $i = (i_1, ..., i_n) \in T^N$. Given that this is the game of common interest, sender and receiver have identical preferences, so $U_S \equiv U_R \equiv U$. The payoff in this game is $U(t, w, i) = -L(t, w, i)$, where $L$ is the expected loss of the players, that they want to minimize, $L : T \times \Sigma \times T^N \to \mathbb{R}$, and is given by $L(t, w, i) = \int_T l(||t - i_{w(t)}||)F(dt)$. $T \subseteq \mathbb{R}^K$ is the type set which is convex, compact and with non-empty interior, $K \geq 1$. The probability of types is described by an atomless distribution $F$ on $T$ which is strictly positive and with continuous density $f : T \to \mathbb{R}_+$.

3.2 Cheap-Talk Equilibria

First, it is assumed that players meet before the game starts to have a cheap talk. We are looking for an equilibrium that they can achieve in cooperation.
3.2. Cheap-Talk Equilibria

Both players aim to understand each other the best they can. They want that type $t$ and its interpretation $i$ be as close as possible to each other, by minimizing the expected loss: $L(t, w, i) = \int l(||t - i_w||)F(dt)$. Equilibria that are minimizing this loss are called the efficient languages.

In the finite case, when the number of types is equal to the number of words, in separating equilibria, this loss would be zero. Since we are here in the situation when the type set is much bigger than the signal set, the best that players can achieve is equilibrium that is "as separating as possible".

Sender has to send one signal that will represent many different types. Since we assumed that the type set is continuum, this means that sender should make partition of the type set, say $(C_k)_{k=1,...,n}$, and then whenever his type $t$ belongs to the category $C_k$, he should say word $w_k$.

Figure 1.3 shows the optimal way of making partition. Assume that there are eight "prototypes", $(i_k)_{k=1,...,8}$, that receiver can understand. Now, the sender wants the prototype to be as close to his type as possible. Therefore, he will use Voronoi tessellation with centers in prototypes for his optimal partition as shown in the second picture of the Figure 1.3. In the same picture, the closest interpretation to the given type $t$ is $i_1$, so the sender's type belongs to the cell $C1$ and he says a word $w_1$.

![Figure 3.1: Sender's action: Voronoi tessellation of the type space](image)

What is the reader's best answer after hearing $w_1$? The receiver knows that the type of the sender belongs to the category $C_1$ as shown in the Figure.
3.2. Cheap-Talk Equilibria

but it can be anywhere in that cell. Therefore the receiver’s answer is the average type from the cell $C_1$. The average type is computed according to a given distribution $F$ of the type set.

![Diagram](image)

Figure 3.2: Receiver is choosing an action

This illustration gives us the idea of how the players should choose their optimal strategies. Since the game that we are analyzing is the cheap-talk game, we are looking for the perfect Bayesian equilibrium.

Given that we are solving the communication game, any pair of strategies $(w, i)$ that consists of a measurable mapping $w : T \to W$ (the signaling strategy), and points $i \in T^N$ (the interpretations) will be called a language. The perfect Bayesian equilibrium will be called Voronoi language since the sender’s strategies are forming the Voronoi cells. In the paper [Frank Riedel 2009] it is proven that Voronoi languages exist.

Analysis is restricted to pure strategies and that is reasonable, given that mixed strategies in the communication games lead to the babbling equilibria in which there is no information transmission. Also, it is shown that mixing is not efficient, since it induces a bigger expected loss.

We say that language $(w, i)$ has a full vocabulary if range $w = W$.

**Lemma 3.2.1.** Efficient languages $(w^*, i^*)$ have full vocabulary and interpretations $i^*_k$ are pairwise distinct.

*Proof.* Let $(w, i)$ be an efficient language. We can identify $w$ with the partition $(C_j)_{j=1,...,N}$, given by $C_j = \{ t \in T | w(t) = t \}$. 
Let \((i_j)_{j=1,...,n}\) be a pure strategy for the receiver. Given this strategy, sender can optimally choose a word that leads to an interpretation that is the closest to sender’s type, i.e. \(w(t) \in \arg \min_{j=1,...,N} ||t-i_j||\). The interpretation is not necessarily unique, therefore we always choose the index with the smallest subscript: \(C_j = \{t \in T| k \text{ is the smallest number in } \arg \min_{j=1,...,N} ||t-i_j||\}\).

\(C_k\) is either a convex cell (intersection of convex polyhedra with the type space \(T\)) or empty. Suppose that \(C_N = \emptyset\), meaning that \(w_N\) is not used. We want to prove that the language where this word is not used, has higher average loss than the language with full vocabulary. And if we would split some other non-empty cell into two smaller ones and for both of them if we use different words, we would get a better language.

By definition, word \(w_1\) is used with some positive probability, i.e. set \(C_1\) has positive mass with respect to \(F\). As \(F\) is atomless, we can find two disjoint, convex, non-empty sets \(A_1, A_2\) with \(A_1 \cup A_2 = A\). Let \(j_1 \in T\) be a minimizer of \(\int_{A_1} l(||t-j||)F(dt)\), and \(j_N\) minimizer of \(\int_{A_N} l(||t-j||)F(dt)\). Given that \(l\) is convex, the minimizers are uniquely defined. Furthermore, \(j_1 \neq j_N\) because the minimizers are in the interiors of \(A_1\) and \(A_2\) respectively.

Set \(j_k = i_k\) for \(k = 2,...,N - 1\). Moreover, set \(v(t) = w_1\) for \(t \in A_1\) and \(w(t) = w_N\) for \(t \in A_N\), and \(v(t) = w(t)\) elsewhere. We claim that \((v,j)\) is a better language than \((w,i)\):

\[
L(v,j) - L(w,i) = \int_{A_1} (l(||t-j_1||) - l(||t-i_1||)) + \int_{A_N} (l(||t-j_N||) - l(||t-i_1||)) > 0.
\]

The last inequality comes from the fact that \(j_1\) and \(j_N\) minimize the loss over the sets \(A_1\) and \(A_N\) and \(j_1 \neq i_1\) and \(j_N \neq i_1\).

Now we will show that all interpretations \((i_k)\) are pairwise distinct. Given that the sender is using a partition \(C_k\) of a convex set with positive measure, the optimal interpretation for word \(w_k\) is the "prototype" \(i_k\) that minimizes \(\int_{C_k} l(||t-j||)F(dt)\) for \(j \in T\). As \(C_k\) is convex and \(F\) atomless, the minimizer lies in the interior of the set \(C_k\). In particular, all interpretations \((i_k)\) are pairwise distinct for an efficient language. \(\square\)
As already mentioned, the sender is using Voronoi tessellation of which each cell has a positive measure and is a convex set. Convexity follows from the fact that cells are intersections of convex polyhedra and the type space, as mentioned in the previous proof. It was also mentioned in the previous proof, that the receiver should minimize the expected loss in the cell that contains sender’s type. That minimizer is called the Bayesian estimator. For convex set $C \subset T$ with positive measure, we call $b(C) = \arg \min_{i \in C} \int_C l(||t - i||) F(dt)$ the Bayesian estimator conditional on $C$

This is summarized in the following lemma.

**Lemma 3.2.2.** In the efficient languages $(w^*, i^*)$

1. the sender uses a Voronoi tessellation corresponding to $i^*$, i.e. $F$-almost everywhere $w^*(t) = \arg \min_{j=1,...,N} ||t - i_j^*||$;

2. the receiver uses the best interpretation of the partition induced by $w^*$, i.e. $i_k = b(C^*)$ for $C_k^* = \{t \in T : w^*(t) = w_k\}$.

Sender’s strategy, given in the previous lemma, is not uniquely defined for those types $t$ which are equally far from two or three different interpretations, so on the boundaries of Voronoi cells. Nevertheless a set that contains such points is a null set, thus we can disregard it, and without loss of generality, take the word with smallest index in this case.

A language $(w, i)$ that consists of a Voronoi tessellation for the sender and an Bayesian estimator interpretation for the receiver, both given in the previous lemma, is called a Voronoi language. Now we can give more consistent characterization of the efficient languages.

**Theorem 3.2.3.** Efficient languages are Voronoi languages with full vocabulary.

The opposite direction of the theorem does not hold, therefore it happens that not all Voronoi languages with full vocabulary are efficient. This is shown by the following counter example.
Example 3.1

Let us consider the unit square $[0,1]^2$, with the uniform distribution, quadratic loss $l(d) = d^2$, and two words $W = \{w_1, w_2\}$. In the Figure 3.3, there are presented two different trapezoidal Voronoi tessellations of the unit square. Figure 3.4 gives the border cases, horizontal and diagonal partitions. By the requirements of the Voronoi languages the centers of Voronoi cells and Bayesian estimators should coincide. Trapezoidal tessellations do no face this requirement, as shown in the second picture of Figure 3.3. Red dots are the Voronoi generators, while the black dots denote the Bayesian estimators. Only border cases do satisfy this requirement. Therefore, we have two Voronoi languages with full vocabulary, horizontal and diagonal. Nevertheless, the diagonal language is not efficient because it has a bigger average loss:

$$L_D(x, y) = \int_0^1 \int_{1-y}^0 ((x - 1/3)^2 + (y - 1/3)) \, dx \, dy \simeq 0.111,$$
then the horizontal language

$$L_D(x, y) = \int_0^1 \int_{1/2}^{1/2} ((x - 1/4)^2 + (y - 1/2)) \, dx \, dy \simeq 0.104.$$

Figure 3.3: Voronoi tessellation of the unit square (Riedel 2009)

3.3 Nash Equilibria

What if the players do not have a cheap talk before they play a game? Then the players have to guess what the opponent will play. Let us assume that
3.3. Nash Equilibria

Figure 3.4: Horizontal and diagonal Voronoi tessellation (Riedel 2009)

they are rational. Therefore, we will do the analysis of Nash equilibrium of this game. We are looking for a strategy that is a best response to itself. A strategy is called a strict Nash equilibrium if it is a unique best response to itself.

There are many Nash equilibria in every signaling game, but here we will only concentrate on pure strategy strict Nash equilibria. We know that mixing leads to the babbling equilibria where is no information transmission. The best replies for sender and receiver coincide with the requirements of the Voronoi languages. Given that the sender is indifferent on the borders of the Voronoi cells, and that the set of "indifference" points is null-set, the sender’s best response is F-almost surely unique.

**Theorem 3.3.1.** Every Voronoi language with full vocabulary is a strict Nash equilibrium and vice versa.

**Proof.** Since the efficient languages are minimizing the expected loss of the players, so \( L(w^*, i^*) \leq L(w, i) \) and \( L(w^*, i^*) \leq L(w, i^*) \) holds, for all \( w \in W, w \neq w^* \) and all \( i \in T^N, i \neq i^* \). That is exactly the requirement for Nash equilibrium (if we put \( U(w, i) = -L(w, i) \)). Strict inequality \( U(w^*, i^*) > U(w^*, i) \) comes from the fact that Bayesian estimator is uniquely determined. Uniqueness follows from convexity of loss function, \( l(||t - i||) \). Other strict inequality \( U(w^*, i^*) > U(w, i^*) \) follows from the fact that the sender’s Voronoi tessellation is F-a.s unique best reply.
Full vocabulary in the Nash equilibrium is a necessary condition because if some word is not used, say $w_N$, than the receiver is indifferent between all interpretations for word $w_N$, and thus the best reply is not unique.

As mentioned in the previous section, it may happen that not all Voronoi languages with full vocabulary are efficient, which here means that even such a strong concept as strict Nash equilibrium does not imply efficiency.
Evolutionary Game Theory

"The most fascinating game that evolution plays is human language." - Martin Nowak

So far we assumed that the players were rational, that they were choosing their strategies, and in equilibrium, that they could not do better. From now on, we allow players to be irrational. Given that, the equilibrium concepts of game theory work only under rationality condition. We resort to using concepts from evolutionary game theory which are relevant even when the players are not reasonable.

We assume that instead of a set of players we have infinite populations, from which we draw a set of players to play a game. Instead of payoff functions, we have fitness which measures the reproductive success. Players do not choose their strategies, instead, their strategies are rather preprogramed by their type. In equilibrium, small mutation in the population can not survive.

After summarizing necessary concepts of evolutionary game theory, there will be given analysis of stability of the Voronoi languages proven in [Frank Riedel 2009].

4.1 Evolutionary dynamics

4.1.1 Discrete strategy space

Evolutionary dynamics is giving us the answer on how to measure the growth of the population or its selection process. Consider the symmetric game with two players. Their sets of strategies are identical $S_1 = S_2 = S$, as well as payoffs $u_1(s_1, s_2) = u_2(s_1, s_2)$. If the strategy set is finite, the payoffs are summarized in the $n \times n$ matrix $A$, where $n$ is the number of
4.1. Evolutionary dynamics

types (strategies). Mixed strategy is given by the vector \( p = (p_1, ..., p_n)^T, \) 
\[ \sum_{i=1}^{n} p_i + \ldots + p_n = 1, \] 
which belongs to the simplex \( S_n \) with the base vectors \( e_i = (0, ..., i, ..., 0)^T, i = 1, ..., n. \)

As already mentioned, the observed population has \( n \) types. Let \( x_i \) be the frequency of the type \( i \). We assume that the population is infinitely large and that \( x_i \) are differentiable functions of time \( t \). The state of the population is denoted by \( x \in S_n, x(t) = (x_1(t), ..., x_n(t))^T \). The evolutionary success of the type \( i \) is given with its growth rate \( \frac{\dot{x}_i(t)}{x_i(t)} \). The expected payoff for the individual \( i \) is \( e_iAx(t) \), and in the language of evolutionary game theory it is called fitness. The average payoff in the population, or average fitness, is given by \( x(t)^T Ax(t) \). The growth rate can be modeled as:

\[
\frac{\dot{x}_i(t)}{x_i(t)} = e_iAx(t).
\]  
(4.1)

We can normalize this term and get population shares: \( p(i) := \frac{x_i(t)}{\sum_{j=1}^{n} x_j(t)} \).

This leads to the replicator dynamics equation:

\[
\dot{p}_i(t) = p_i(t)(e_iAp(t) - p(t)Ap(t)), i = 1, ..., n.
\]  
(4.2)

Therefore, the evolutionary success is expressed as the difference between the fitness of the type \( i \) and the average fitness of the whole population.

Unit simplex (the set of probability vectors) \( S_n = \{ p \in \mathbb{R}^n : p_i \geq 0, \sum_{i=0}^{n} = 1 \} \) is invariant under replicator dynamics, which means that if \( p(0) \in S_n \) then \( p(t) \in S_n \) for every \( t > 0 \). From now on we will consider only restriction of the replicator dynamics equation on this simplex.

In the following, we give definitions of the basic concepts: what do we consider an equilibrium of a dynamical system, when do we say that it is stable, and when asymptotically stable.

**Definition 2.** Let \( \dot{p}_i(t) = f_i(p, t), i = 1, ..., n \) be a dynamical system defined on \( K \subseteq \mathbb{R}^n \). \( p^* \in K \) is called an equilibrium (stationary state) if all trajectories \( (p(t))_{t \geq 0} \) with \( p(0) = p^* \) satisfy \( p(t) = p^* \) for all \( t \geq 0 \).

By solving equation (4.2) we get the stable states of the replicator dynamics, so \( p_i(t)(e_iAp(t) - p(t)Ap(t)) = 0 \) for every \( i = 1, ..., n \). There are two solutions:
4.1. Evolutionary dynamics

1. $p_i(t) = 0, i = 1, ..., n$, gives us that the corners $p^* = e_i$ of the simplex $S_n$ are always solutions. This implies that all homogenous populations are equilibria;

2. $e_i Ap(t) = p(t) Ap(t), i = 1, ..., n$ which will give us the interior points from the simplex $S_n$.

Definition 3. Let $p^*$ be an equilibrium of the dynamical system, $p^*$ is stable if for every $\epsilon > 0$, there is $\delta > 0$ such that for all $||p(0) - p^*|| < \delta$ we have that $||p(t) - p^*|| < \epsilon$. $p^*$ is called asymptotically stable if it is stable and if there exists $\epsilon > 0$ such that $\lim_{t \to \inf} p(t) = p^*$ for $||p(0) - p^*|| < \epsilon$.

Let us now see the relations between Nash equilibria and the equilibria of replicator dynamics.

Theorem 4.1.1. 1. $(p^*, p^*)$ is a Nash equilibrium in the game with strategy sets $S_1 = S_2 = \{s_i \in S_i : p_i^* > 0\} = \text{supp}(p^*)$ if and only if $p^*$ is an equilibrium of replicator dynamics;

2. if $p^*$ is stable, then $(p^*, p^*)$ is Nash equilibrium;

3. if $(p^*, p^*)$ is strict Nash equilibrium, then $p^*$ is asymptotically stable.

There are few ways of testing whether the equilibrium is (asymptotically) stable. Here I present only one using the Lyapunov function. I decided to use this one, given that, as will be shown later, it is also suitable for checking stability conditions of Voronoi languages.

Definition 4. Let $U \subseteq K$ be an open set and $p^* \in U$ an equilibrium. Continuous function $L : U \to \mathbb{R}$ is called (strict) Lyapunov function if:

1. $L(p) > 0$ for $p \in U, p \neq p^*, L(p^*) = 0$

2. for every solution $(p(t))$ of (4.1) with $p(t) \in U$ for $t \in [0, T]$, we have $t \to L(p(t))$ (strictly) decreasing on $[0, T]$.

Theorem 4.1.2. Suppose there exists a (strict) Lyapunov function (for $p^*$). Then $p^*$ is (asymptotically) stable.
4.1.2 Continuous strategy space

From now on the strategy set, $S$, is a subset of the real space $\mathbb{R}$. A population of agents is described by a probability measure $P$ on the measure space $(S, \mathcal{B})$, where $\mathcal{B}$ denotes the Borel $\sigma$-algebra on $S$. $\Delta S$ is the set of all populations on $S$ (mixed strategies). A payoff function is given by $f : S \times S \to \mathbb{R}$ and is assumed to be bounded and Borel measurable.

The idea of the dynamic stability is to distinguish which strategies are close enough to the equilibrium in order to keep them. The biggest problem is how to define the "closeness" to the original equilibrium. Oechssler and Riedel in [Oechssler 2002] claimed that the most suitable way to measure the differences between populations is the weak topology$^3$.

They used Levy-Perhorov metric to measure the distance between the populations, since convergence of measures in this metric is equivalent to weak convergence of measures. It is defined as

$$
\rho(P, Q) := \inf \{\varepsilon > 0 : Q(B) \leq P(B^\varepsilon) + \varepsilon \text{ and } P(B) \leq Q(B^\varepsilon) + \varepsilon, \forall B \in \mathcal{B}\},
$$

where $B^\varepsilon = \{x : \exists y \in B, |y - x| < \varepsilon\}$, and $Q$ probability measure on $(S, \mathcal{B})$. So, $P_n$ converges weakly to $P$ if and only if $\rho(P_n, P) \to 0$.

The average payoff of the population $P$ against population $Q$ is given by

$$
E(P, Q) = \int_S \int_S f(x, y)Q(dy)P(dx).
$$

It is assumed that it is linear in both $P$ and $Q$, and since $f$ is continuous, $E(P, Q)$ is also continuous in weak topology.

Evolutionary dynamics are regular if $\dot{P}(S) = 0$ and $\dot{P}(B) = 0$ for all $B \in \mathcal{B}$ with $P(B) = 0$.

**Definition 5.** A regular dynamic is called payoff monotonic if for all $B$ and $B' \in \mathcal{B}$ with $P(B), P(B') > 0$, $\dot{P}(B) > \dot{P}(B')$ if and only if

$$
\frac{1}{P(B)} \int_B E(\delta_x, P)P(dx) > \frac{1}{P(B')} \int_{B'} E(\delta_x, P)P(dx).
$$

$^3$We say that a sequence of probability measures on the measure space $(S, \mathcal{B})$, $P_n$, converges weakly to the probability measure $P$ if $\int_S f dP_n \to \int_S f dP$ for every bounded and continuous real function $f$. 
4.2. Static evolutionary concepts

This means that, dynamics are monotonic if the set of types with higher average fitness has higher growth rates.

Let $P^*$ be an equilibrium of the dynamics, that is $P^*(B) = 0$ for all Borel subsets $B$ on $S$. $P^*$ is called Lyapunov stable if for all $\varepsilon > 0$ there exists $\delta > 0$ such that for all initial populations $Q(0)$ with $\rho(Q(0), P^*) < \delta$ we have $\rho(Q(t), P^*) < \varepsilon$ for all $t > 0$.

The generalized replicator dynamics (for the continuous case) are given by (see [Oechssler 2000]):

$$\dot{P}(t)(B) = \int_B \sigma(x, P(t)) P(t)(dx).$$

(4.4)

Here

$$\sigma(x, P) := E(\delta_x, P) - E(P, P)$$

denotes the differential fitness of pure strategy $x$ when matched against population $P$. In other words, that indicates the difference between the payoff of strategy $x$ and the average population payoff. When $B = x$, (4.4) gives the formula of replicator dynamics for the finite case.

4.2 Static evolutionary concepts

The first stability concept, evolutionary stable strategy (ESS), goes back to Maynard Smith and Price, and it ensures dynamical stability for replicator dynamics (4.2) in the games with a finite number of pure strategies.

Definition 6. For all mutant populations $Q$ there exists invasion barrier $\varepsilon > 0$ such that the original population $P$ does better against the mixed population $(1 - \eta)P + \eta Q$ than $Q$ for all $\eta \leq \varepsilon$.

When the strategy space is continuous, the ESS is not any longer sufficient. There were proposed few different stability concepts for continuous strategy space, such as CSS (Eshel), uninvadability (Bonze, Poetscher), and NIS (Apaloo), but none of them ensures dynamical stability in the weak topology.
4.3. Stability of Voronoi languages

Then, Oechssler and Riedel in [Oechssler 2002] established a new stability concept, Evolutionary robustness ($ER$). Evolutionary robustness is generalization of ESS for the continuous models with weak topology, and in the finite strategy space, two concepts coincide.

**Definition 7.** A population $P^* \in \Delta S$ is evolutionary robust if there exists $\varepsilon > 0$ such that for all $Q \neq P^*$ with $\rho(Q, P^*) < \varepsilon$ we have $E(P^*, Q) > E(Q, Q)$. The supremum of the $\varepsilon$ with this property is called the invasion barrier of $P^*$.

In the finite case, ESS does not need to exist, and the same holds for continuous case and $ER$.

**Theorem 4.2.1.** Consider a doubly symmetric game with continuous payoff function $f$ and compact strategy set $S$. Let $P^*$ be $ER$. The function $\Lambda(Q) = E(P^*, P^*) - E(Q, Q)$ is a Lyapunov function with respect to the replicator dynamics. Thus, $P^*$ is Lyapunov stable.

Proof of this Theorem is given in [Oechssler 2002]. Parts of this proof will be used for proving the stability of Voronoi language.

4.3 Stability of Voronoi languages

Let us now analyze the stability properties of the Voronoi languages. For this purpose we need a symmetric game. However, our game is signaling game and therefore non-symmetric. To "symmetrize" the game, we suppose that the agents are equally often in the role of receiver and sender. This is very reasonable assumption, since in real life there are no people that are only senders or only receivers, they are changing their roles in different situations.

Every agent chooses both a sender strategy $v$ or $w \in \Sigma$, and a receiver strategy $i$ and respectively $j \in T^N$. Then the expected loss of the player using the language $(v, i)$, who is meeting the player using the language $(w, j)$ is

$$\Lambda((v, i), (w, j)) = 1/2(L(v, j) + L(w, i)).$$ (4.5)

A population is described by a probability distribution $P(dw, di)$ over the strategy set $\Gamma := \Sigma \times T^N$ of the symmetrized game. For two such distributions
4.3. Stability of Voronoi languages

We can extend the symmetrized loss function in the following way:

\[ \Lambda(P, Q) = \int_{\Gamma} \int_{\Gamma} \Lambda((v, i)(w, j))P(dw, di)Q(dw, dj). \]  

(4.6)

Now we can apply the results from the previous chapter, when the average payoff is \( E(P, Q) = -\Lambda(P, Q) \).

It is proven that the fundamental law of natural selection holds for our symmetrized payoff function, so it is a Lyapunov function for the replicator dynamics. Also, the loss function is continuous in weak topology.

**Lemma 4.3.1.** Evolutionary robust languages are strict local optima.

**Proof.** By the definition of \( \mathcal{ER} \), we have for \( Q \) close to \( P^* \)

\[
\Lambda(Q) = E(P^*, P^*) - E(Q, Q) \\
= E(P^*, P^*) - E(P^*, Q) - E(P^*, Q) - E(Q, Q) \quad \text{where the least equality follows from symmetry of } f \text{ and the last inequality from the fact that every } \mathcal{ER} \text{ is a symmetric Nash equilibrium. } P^* \text{ is a strict local minimum of } \Lambda(Q, Q).
\]

At this point, all the requirements for the Theorem 4.2.1 are satisfied. We have doubly symmetric game with the continuous loss function and a compact strategy set, so if the strategy \( P^* \) is evolutionary robust, then it is Lyapunov stable. The following theorem summarizes this.

**Theorem 4.3.2.** Locally optimal languages are Lyapunov stable with respect to replicator (more general, regular, payoff-monotone) dynamics.

For the finite strategy space, strict Nash equilibrium is implying ESS, however when the strategy space is continuum, strict Nash does not imply \( \mathcal{ER} \). This means that, even such a strong equilibrium concept does not ensure dynamical stability and that the Voronoi languages can be unstable. This is proven in the paper [Frank Riedel 2009] by the same counter example analyzed in the previous chapter. Here also the diagonal language appears to be unstable. Nevertheless the efficient horizontal language is stable.
4.3. Stability of Voronoi languages

It also happens that not every stable equilibrium is efficient. Some inefficient languages can be local optima, i.e. Lyapunov stable. Therefore, inefficiency can appear even under a strong condition such as $\mathcal{E}\mathcal{R}$. 
Chapter 5

The Algorithm and Simulations

5.1 The Algorithm

Given that it is analytically very hard to analyze equilibria properties of Voronoi languages that have more than three words and even harder to examine their stability properties, we are obliged to rely on simulations. Here is presented a simple algorithm that is selecting stable languages from a variety of Voronoi languages. The algorithm is taken from the article [Frank Riedel 2009].

In the observed article, there was only presented program that searches for stable languages when the type set is uniformly distributed. In this chapter are presented results of the program which also selects stable languages when the type set is not uniformly distributed.

Nevertheless the improved program presented here, is still limited, since it gives only languages with the two-dimensional type set. Therefore in the further research one can extend the model to work for n-dimensionaly type set, $n > 2$. In that case, one can expect to get higher-dimensional Voronoi diagrams. Also, one can use some other metric different than quadratic which was used here.

The algorithm has three main steps:

**First step** Program chooses randomly, according to a given distribution of the type set, $N$ initial values. In Figure 5.1 (a), there are randomly chosen three initial values.

These initial values are generators of the Voronoi tessellation of the type space, but it is very probable that in this partition centers of the cells do not coincide with the Bayesian estimators, which is a necessary condition for the
Voronoi languages. The way in which we obtain that is given in the next two steps. Those two steps are repeated finitely many times.

**Second step** In every iteration program randomly chooses (according to given distribution) finitely many types from the type space, and then assigns each type with its closest interpretation. As it is shown in the second panel of the Figure 5.1, closest interpretation to red dots is $i_1$, to green $i_2$, and purple $i_3$.

**Third step** In the last step, program computes arithmetic mean of the chosen types in each cell, and than those arithmetic means are becoming initial values for the next iteration. That is shown in the third panel of the Figure 5.1, old and new centers of each cell, and the new space tessellation. In this way we are moving generators of the Voronoi cells until they coincide with the Bayesian estimators.

It is very important to use the same distribution of the type set for updating the centers of Voronoi cells, because otherwise the final language would not follow the initial distribution and would become some other language.

For the case of two words, the stable language has quadratic form, as well as the language with four words, and one would expect that every language with $n^2$ words, $n > 2$, would follow the same rule as shown in the Figure 5.2. Nevertheless, it happens that the quadratic structure does not survive evolution. The stable structure is hexagonal. It can be seen from the Figure 5.3.
5.2 Simulations - Different types of distributions

As already discussed, the uniformly distributed type set is not very realistic for applications, one would need the possibility to work with some other types of distributions. In this section, we present some representative pictures for differently distributed type sets.

The type set throughout this chapter is assumed to be two-dimensional Euclidian space, so therefore when we say that it (type set) follows some distribution, we mean that both random variables follow that particular distribution, and that they are independent. Situation when distribution has different parameters for those two random variables is also analyzed.

First, we start with the picture of the uniformly distributed types because it is very clear that evolution leads to hexagonal tessellation of the type space. Figures 5.4 and 5.5 give evolution of the uniformly distributed language with 120 words. In the first picture we can see a Voronoi tessellation
5.2. Simulations - Different types of distributions

where the generators are randomly drawn from the uniformly distributed type set. Second picture shows that after only 30 iterations, the cells are already getting hexagonal structure. In the last two pictures, we can see that there are not big differences between language after 100 iterations and after 1000. This shows that observed program selects Voronoi languages with full vocabulary very fast, among a variety of languages.

Now, we do the analysis of languages with non-uniformly distributed type set. In this paper are analyzed languages with normally, beta, and exponentially distributed type sets.

Figure 5.6 gives us the exponentially distributed Voronoi language
5.2. Simulations - Different types of distributions

Figure 5.6: Exponential distribution, $X \sim \mathcal{E}(8)$, $Y \sim \mathcal{E}(8)$, 100 words, 100,000 iterations

Figure 5.7: Exponential distribution, $X \sim \mathcal{E}(3)$, $Y \sim \mathcal{E}(8)$, 100 words, 100,000 iterations

with full vocabulary, where both random variables are following the same distribution, $X, Y \sim \mathcal{E}(8)$. Figure 5.7 presents tessellation of asymmetrically distributed types. Distributions have different parameters and $Y$ follows much steeper exponential distribution than $X$. We can notice here that regions with lower weight have larger cells and that smaller cells are in the regions with higher mass.

In the next chapter we will see that exponential distribution is very important for analyzing if the Voronoi languages are following the Zipf’s law.
5.3. Applications

Figures 5.8 and 5.9 show Voronoi languages whose type sets follow beta distributions with different parameters. In the Figure 5.10 we can see the tessellation when the type set is normally distributed.

5.3 Applications

If we would forget about the roles of players and think about Voronoi equilibria only as the two criteria that should be satisfied, first that the optimal partition of some space is Voronoi tessellation and second that generators of
Voronoi cells should coincide with the centers of mass (average types), than we are opening our algorithm to the very broad range of applications.

Given that Voronoi tessellations appear very often in nature, and the most interesting ones are so called centroidal Voronoi tessellations, which means that the centers of mass of Voronoi cells coincide with the generators. This definition has similar requirements as our Voronoi languages.

Therefore our algorithm and program could be used in many other situations different than languages. The paper by [Qiang Du 1999] presents many different computational programs and applications of centroidal Voronoi languages.
5.3. Applications

One of the applications could be an optimal placement of resources (location problems). For example, the optimal placement of the mailboxes in a city. If we assume that the users would use only the mailbox that is the closest to their home, that the cost of using it is the distance from the home to the mailbox, and that the total cost is the sum of all user’s costs, then the optimal location would be the one that is minimizing the total cost.

Very interesting application is data compression in image processing. Pixels of every image have their RGB representation (the intensity of red, blue and green color). The idea how to make compression is by replacing the color (say 24-bit color) of each pixel by one from the smaller set of colors (8-bit). Therefore the number of pixels remains the same, and we are reducing the amount of information associated with each pixel. By the similar algorithm, as ours, given in [Qiang Du 1999], the authors showed that "Voronoi" compression gives better results than Monte Carlo method.

One example from the nature is territorial behavior of animals. Many different animal species use Voronoi tesselations to separate their territories. Fish Tilapia mossambica from the desire to be as far from its neighbors as possible, uses the uniformly distributed Voronoi tessellation as shown in the Figure 5.12 and picks the center of the cell as the best place for breeding.

There are many other applications such as optimal quadrature rules, optimal clustering, and cell division, among others.
In this chapter we examine if our "artificial" Voronoi languages have some properties of "live" languages. One of the interesting characteristics of all natural languages, that can be tested, is that they all satisfy Zipf's law.

Georg K. Zipf [Zipf 1972] found that in every corpus of natural language utterances, there exists some kind of a "vocabulary balance". Zipf's law states that in every language the most frequent word occurs twice as often as the next most common word, the third two times less than second, and so on. This means that there is some relation between the absolute frequency and the rank of each word in some corpus. And it is:

\[ f = \frac{C}{r^a}, \]

where \( r \) is the rank of the word, \( f \) its frequency, \( C \) constant, and \( a \) parameter close to 1. For presenting this law graphically, the best way is to take logarithm of both sides \( \log_{10} f = \log_{10} C - a \log_{10} r \) and then plot \( \log_{10} r \) on \( x \)-axis, \( \log_{10} f \) on \( y \)-axis and if some data follow the Zipf's law, one should get straight line that has slope -1.

For example, in English language the most common word is "the", on the second place is word "of", then follows "and",... Figure 6.1 shows loglog graph where on \( x \)-axis is rank of the word and on \( y \)-axis its frequency. It is obvious that the best approximation of this graph is indeed the straight line with the slope of -1.

If we want to connect this with our model, we can think about frequencies of words as weights of our cells, since the weights or volumes are actually probabilities for particular word to be chosen. If the volume is bigger, assigned word has greater probability to be chosen and will occur more frequently.
In the more formal way, the set of words (signals) is $W = \{w_1, ..., w_m\}$. We can look at text as a sequence $S_L$ of words, $S = \{s_1, ..., s_L\}$, where $s_i \in W$, and $i = 1, ..., L$, $N_j$ is the number of occurrences of the word $w_j$, where $j = 1, ..., m$ in a text $S_L$, and $p_L(j) = \frac{N_j}{L}$, denotes the frequency (probability) of word $w_j$ in the text $S_L$, so $p(1) + ... + p(m) = 1$. We can assume the following order: $p(1) \geq p(2) \geq ... \geq p(m)$. Now the Zipf's law can be written in the form:

$$p(i) = \frac{p(1)}{i^a}$$

(6.2)

where $a$ is close to 1 and $p(1)$ is the probability of the most frequent word. In the logarithmic form it looks as follows:

$$\log_{10} p(i) = \log_{10} p(1) + a \log_{10} i.$$  

(6.3)

We can look at our Voronoi languages in a way that different distributions of the type set give different languages. The problem is how the type set should be distributed in order to satisfy this law. It is obvious that it does not work with uniform distribution, since the weights are very similar, so all words would have similar frequencies. One should think about some much "sharper" distribution. Here are presented results for normal distribution and exponential distribution, given that those are the most interesting cases.

Let us start with a language that has exponentially distributed type set. First observation is that languages that are not stable (maybe not even Voronoi) are following the Zipf’s law better than the stable languages. This may look surprising at first. However, this turns out to make sense since when the language is stable, there are many cells that have the same weights,
Figure 6.2: Exponential, $Exp(1)$, 80 words, slope$=-1.498$

Figure 6.3: Exponential, $Exp(1)$, 80 words, 1000 iterations, slope$=-1.193$
while when the language is not stable, it is more probable that most of the
cells are different. This is shown in the Figures 6.2 and 6.3. So, the non-stable
language (Figure 6.2) is very well approximated with the straight line, but
the slope significantly differs from −1. Therefore we can think of that as the
more general Zipf’s law. The stable language has a better trend, apart from
that it is not well approximated with the straight line (as can be seen in the
Figure 6.3). We see that some "stairs" appeared, those "stairs" indicate that
many cells have the same weight and therefore the Zipf’s law is not satisfied.
One more observation is that in these pictures there is presented language
with 80 words, and when number of words increases the results deviate from
the Zipf’s law. Linear predictors produce higher prediction error when the
number of words taken into account grows larger.

Now, let us focus on languages with normally distributed type set. It
appears that stability of the language does not matter so much. This is
shown in the Figures 6.4 and 6.5. Both stable and non-stable languages are
very well approximated with the straight line up to some threshold value, and
after that both have a "dropping end". Nevertheless many papers from the
literature suggested that it can still be considered as a good approximation.
However, the problem is that the slope is very close to 0 which indicates that
actually weights are very similar.

The general remarks are that, so far, non-stable exponentially distributed
languages are following Zipf’s property much better than all the other ob-
served languages. So far we were exploring the impact of different parameters of observed distributions, number of words and the "degree" of stability (number of iterations) on appearance of the Zipf’s law. The best results that we got are presented in the figures of this chapter. Nevertheless, many different examinations can still be done. One can also use some different approximation for computing the weights of the cells. This can be a starting point for some further research.

The code of the program and the technical explanations are given in the appendix A2.
Chapter 7

Conclusions

This paper analyzes equilibria of the signaling games that have continuum, subset of multidimensional Euclidian space, as a type set and a finite set of signals. It is based on the results of the paper by [Frank Riedel 2009], who analyzed the equilibria that players can achieve in cooperation, called Voronoi languages. They proved that strict Nash equilibria of this game are exactly Voronoi languages. The efficient languages are evolutionary stable, however, it may also happen that some inefficient language survives the evolution.

Given that the problem is very complex and that one can not do the analysis of equilibria for languages that have more than three words, people usually resort to computational approach. The program that was introduced in the cited paper is very limited in the sense that it works only for uniformly distributed type sets of the sender. This thesis presents a modified program written in the programing language C++ that is much more efficient, works for many different distributions, and can be easily extended to work for even wider range of distributions. This can be achieved whenever we have explicitly given quintile function of the desired distribution.

This thesis also presented possible applications of the program, such as location problems, and problems of data compressing, among others.

Finally, it presented the analysis of Zipf’s law property of Voronoi languages. After analyzing normally and exponentially distributed languages we found that the arbitrary chosen exponential languages satisfy the Zipf’s law, while the stable ones do not.
A.1 Code for the Voronoi languages

There is given code (written in the programing language C++) for computing nodes (centers) of each Voronoi cell. After computing the nodes, one can use the second program (written in MatLab) for plotting the Voronoi tessellation.

Voronoi.cpp
#include <iostream>
#include <cstdlib>
#include <math.h>
#include <fstream>
using namespace std;

## Function that returns normally distributed random number

double randn_notrig(double mu=0.0, double sigma=1.0) {
    static bool deviateAvailable=false;
    static float storedDeviate;
    double polar, rsquared, var1, var2;
    if (!deviateAvailable) {
        do {
            var1=2.0*( double(rand())/double(RAND_MAX) ) - 1.0;
            var2=2.0*( double(rand())/double(RAND_MAX) ) - 1.0;
            rsquared=var1*var1+var2*var2;
        } while ( rsquared>=1.0 || rsquared <= 0.0);
        polar=sqrt(-2.0*log(rsquared)/rsquared);
        storedDeviate=var1*polar;
        deviateAvailable=true;
    }
    return var2*polar*sigma + mu;
}
else {
    deviateAvailable=false;
    return storedDeviate*sigma + mu;
}

int main (void) {
    ofstream myfile1;
    ofstream myfile2;
    myfile1.open ("x_coordinate.txt");
    myfile2.open ("y_coordinate.txt");

    int N ; //number of words
    int T ; //number of points in each cell
    int F ; //iterations
    char indicator ;

    cout << "Choose the distribution for type set of the sender: type 'u' for uniform
distribution; type 'e' for exponential distribution; type 'n' for normal distribution;
type 'a' for 'antinormal' distribution: " ;
    cin >> indicator ;

    cout << "Enter the number of words: " ;
    cin >> N ;

    cout << "Enter the total sample size: " ;
    cin >> T ;

    cout << "Enter the number of iterations: " ;
    cin >> F ;

    double x[N] ;
    double y[N] ;
    float lambda = 0.95 ;
    double min = 2 ; //initial minimal distance
    int number = 0 ; //initial number of samples in each tile
    double xold[N] ;
    double yold[N] ;
    double v[T][3] ;
A.1. Code for the Voronoi languages

```c
srand ( time(NULL) ) ;

//uniformly distributed type set:

if ( indicator == 'u' )
{
    for (int i = 1; i < N+1; ++i)
    {
        x[i] = (double)rand() / RAND_MAX ;
        y[i] = (double)rand() / RAND_MAX ;
    }
    for (int i = 1; i < N+1; ++i)
    {
        xold[i] = 0 ;
        yold[i] = 0 ;
    }
    for ( int f = 1; f < F+1; ++f)
    {
        for (int i = 1; i < 4; ++i)
        {
            for (int j = 1; j < T+1; ++j)
            {
                v[j][i] = (double)rand() / RAND_MAX ;
            }
        }
        for (int s = 1; s < T+1; ++s)
        {
            min = 2 ;
            for ( int t = 1; t < N+1; ++t)
            {
                if ( (v[s][1] - x[t])*(v[s][1] - x[t]) + (v[s][2] - y[t])*(v[s][2] - y[t]) < min )
                {
                    v[s][3] = t ;
                    min = (v[s][1] - x[t])*(v[s][1] - x[t]) + (v[s][2] - y[t])*(v[s][2] - y[t]) ;
                }
            }
        }
        for ( int n = 1; n < N+1; ++n)
        {
            number = 0 ;
        }
    }
```
for (int t = 1; t < T+1; ++t) {
    if (v[t][3] == n) {
        number = number + 1;
    }
}
if (number > 0) {
    for (int i = 1; i < N+1; ++i) {
        xold[i] = x[i];
        yold[i] = y[i];
    }
    x[n] = 0;
    y[n] = 0;
    for (int t = 1; t < T+1; ++t) {
        if (v[t][3] == n) {
            x[n] = x[n] + v[t][1];
            y[n] = y[n] + v[t][2];
        }
    }
    x[n] = xold[n]*lambda + (1-lambda)*x[n] / number;
    y[n] = yold[n]*lambda + (1-lambda)*y[n] / number;
}
}

//'antinormally distributed type set:

if (indicator == 'a') {
    for (int i = 1; i < N+1; ++i) {
        x[i] = randn_notrig(0.0, 0.15);
        y[i] = randn_notrig(0.0, 0.15);
    }
}
for (int i = 1; i < N+1; ++i)
{
    xold[i] = 0 ;
    yold[i] = 0 ;
}
for ( int f = 1; f < F+1; ++f)
{
    for (int i = 1; i < 4; ++i)
    {
        for (int j = 1; j < T+1; ++j)
        {
            v[j][i] = randn_notrigio9om o91.j s
        }
    }
    for (int s = 1; s < T+1; ++s)
    {
        min = 2 ;
        for ( int t = 1; t < N+1; ++t)
        {
            if ( (v[s][1] - x[t])* (v[s][1] - x[t]) + (v[s][2] - y[t])* (v[s][2] - y[t]) < min )
            {
                v[s][3] = t ;
                min = (v[s][1] - x[t])* (v[s][1] - x[t]) + (v[s][2] - y[t])* (v[s][2] - y[t]) ;
            }
        }
    }
    for ( int n = 1; n < N+1; n=n+2)
    {
        number = 0 ;
        for ( int t = 1; t < T+1; ++t)
        {
            if ( v[t][3] == n)
            {
                number = number + 1 ;
            }
        }
        if ( number > 0)
        {
            for ( int i = 1; i < N+1; ++i)
            {
            
            }
A.1. Code for the Voronoi languages

```c
xold[i] = x[i] ;
yold[i] = y[i] ;
}
x[n] = 0 ;
y[n] = 0 ;
for ( int t = 1; t < T+1; ++t )
{
    if ( v[t][3] == n )
    {
        x[n] = x[n] + v[t][1] ;
y[n] = y[n] + v[t][2] ;
    }
}
x[n] = xold[n]*lambda + (1-lambda)*x[n] / number ;
x[n+1] = x[n] + 1 ;
y[n] = yold[n]*lambda + (1-lambda)*y[n] / number ;
y[n+1] = y[n] + 1 ;
}
}
}
// normally distributed type set

else if ( indicator == 'n' )
{
    for (int i = 1; i < N+1; ++i)
    {
        x[i] = randn_notrig(0.0, 0.7) ;
y[i] = randn_notrig(0.0, 0.7) ;
    }
for (int i = 1; i < N+1; ++i)
{
    xold[i] = 0 ;
yold[i] = 0 ;
}
for ( int f = 1; f < F+1; ++f)
{
    for (int i = 1; i < 4; ++i)
    {
```
for (int j = 1; j < T+1; ++j)
{
    v[j][i] = randn_notrigio9om o9qj s
}
}
for (int s = 1; s < T+1; ++s)
{
    min = 2;
    for (int t = 1; t < N+1; ++t)
    {
        if ( (v[s][1] - x[t])*v[s][1] - x[t]) + (v[s][2] - y[t])*v[s][2] - y[t]) < min
        {
            v[s][3] = t;
            min = (v[s][1] - x[t])*(v[s][1] - x[t]) + (v[s][2] - y[t])*v[s][2] - y[t])
        }
    }
}
for (int n = 1; n < N+1; ++n)
{
    number = 0;
    for (int t = 1; t < T+1; ++t)
    {
        if ( v[t][3] == n)
        {
            number = number + 1;
        }
    }
    if ( number > 0)
    {
        for (int i = 1; i < N+1; ++i)
        {
            xold[i] = x[i];
            yold[i] = y[i];
        }
        x[n] = 0;
        y[n] = 0;
        for (int t = 1; t < T+1; ++t)
        {
            if ( v[t][3] == n)
            {
            }
x[n] = x[n] + v[t][1] ;
y[n] = y[n] + v[t][2] ;
}
}
x[n] = xold[n]*lambda + (1-lambda)*x[n] / number ;
y[n] = yold[n]*lambda + (1-lambda)*y[n] / number ;
}
}
}

// exponentially distributed type set

ext if ( indicator == 'e' )
{
  double lambda1 ;
  double lambda2 ;
  cout << "Enter the value for parameter lambda (for x-coordinate): " ;
  cin >> lambda1 ;
  cout << "Enter the value for parameter lambda (for y-coordinate): " ;
  cin >> lambda2 ;

  for (int i = 1; i < N+1; ++i)
  {
    x[i] = -log ( 1.0 - (double)rand() / RAND_MAX ) / lambda1 ;
    y[i] = -log ( 1.0 - (double)rand() / RAND_MAX ) / lambda2 ;
  }
  for (int i = 1; i < N+1; ++i)
  {
    xold[i] = 0 ;
    yold[i] = 0 ;
  }
  for ( int f = 1; f < F+1; ++f)
  {
    for (int j = 1; j < T+1; ++j)
    {
      v[j][1] = -log ( 1.0 - (double)rand() / RAND_MAX ) / lambda1 ;
      v[j][2] = -log ( 1.0 - (double)rand() / RAND_MAX ) / lambda2 ;
      v[j][3] = 1 ;
    }
A.1. Code for the Voronoi languages

for (int s = 1; s < T+1; ++s) {
    min = 2;
    for (int t = 1; t < N+1; ++t) {
        if ( (v[s][1] - x[t])*(v[s][1] - x[t]) + (v[s][2] - y[t])* (v[s][2] - y[t]) < min ) {
            v[s][3] = t;
            min = (v[s][1] - x[t])*(v[s][1] - x[t]) + (v[s][2] - y[t])* (v[s][2] - y[t]);
        }
    }
}

for (int n = 1; n < N+1; ++n) {
    number = 0;
    for (int t = 1; t < T+1; ++t) {
        if ( v[t][3] == n ) {
            number = number + 1;
        }
    }
    if ( number > 0 ) {
        for (int i = 1; i < N+1; ++i) {
            xold[i] = x[i];
            yold[i] = y[i];
        }
        x[n] = 0;
        y[n] = 0;
        for (int t = 1; t < T+1; ++t) {
            if ( v[t][3] == n ) {
                x[n] = x[n] + v[t][1];
                y[n] = y[n] + v[t][2];
            }
        }
    }
}
A.2. Code for the Zipf’s law

\[
\begin{align*}
x[n] &= \text{xold}[n] \ast \text{lambda} + (1-\text{lambda}) \ast x[n] \text{ / number} ; \\
y[n] &= \text{yold}[n] \ast \text{lambda} + (1-\text{lambda}) \ast y[n] \text{ / number} ;
\end{align*}
\]

After building the sequences of Voronoi centers, one can plot Voronoi tessellation. In the following there is given simple program for plotting, first import x_coordinate and y_coordinate, which are saved in the same directory where the C++ code is.

```
Voronoi_plot.m
x = x_coordinate;
y = y_coordinate;
voronoi(x,y)
box on
axis([0 1 0 1])
```

\section*{A.2 Code for the Zipf’s law}

```
Zipf_exponential.m
x = x_coordinate;
y = y_coordinate;
```
A.2. Code for the Zipf’s law

```matlab
% parameters of the distributions
lambda1 = 1 ;
lambda2 = 1 ;

% compute the weights
for i = 1 : size(c ,1)
    ind = c{i}’;
    tess_area(i,1) = polyarea( v(ind,1) , v(ind,2) ) ;
    z(i) = tess_area(i,1) * lambda1 * lambda2 * exp( -x(i)*lambda1 -y(i)*lambda2 ) ;
end

% sort the weights in the decreasing order
B = sort(z, ’descend’);
N = size(B ,2) - 11;
for k = 1 : N
    B2(k) = B(k+11);
end
for j = 1 : size(B2 ,2)
    D(j) = j ;
end
B1 = log10(B2);
D1 = log10(D);
fit1 = polyfit(D1,B1,1) ;
f = polyval(fit1, D1) ;
plot(D1, B1, '.', D1, f, '-r', D1, 0 -D1, ’g’) ;
xlabel(’log10 rank’) ;
ylabel(’log10 frequency’) ;
legend(’data’, ’regresion’, ’Zipfs law’, 0) ;
```
Bibliography


Bibliography


Declaration

Hereewith I affirm that I have written this thesis on my own. I did not enlist unlawful assistance of someone else. Cited sources of literature are perceptibly marked and listed at the end of this thesis. The work was not submitted previously in same or similar form to another examination committee and was not yet published.

Bielefeld, Germany on 31 August 2010