



## Second part report

### DISCRETE OPTIMIZATION

---

**Fabrice Servais** (S111093)  
**Laurent Vanosmael** (S114351)

Masters in Computer Science and Engineering - 1<sup>st</sup> year

December 7, 2014

# 1 Introduction

In this report, we'll first talk about the data and the model that have been used. After that, there will be an explanation on the method that has been used. Finally, we'll present the results that we obtained and discuss some improvements.

# 2 Data

The data listed below are the ones from the Integer Model that we have used in the first report.

- $sl_k = \begin{cases} 1 & \text{if the key } k \text{ is on the left side,} \\ 0 & \text{otherwise.} \end{cases}$
- $sr_k = \begin{cases} 1 & \text{if the key } k \text{ is on the right side,} \\ 0 & \text{otherwise.} \end{cases}$
- $fr_l$  : probability of the apparition of the letter  $l$  in the considered language.
- $vl = \begin{cases} 1 & \text{if the letter } l \text{ is a vowel,} \\ 0 & \text{otherwise.} \end{cases}$
- $V$  : number of vowels in the language.
- $w_{i,j}$  : probability that the letter  $j$  follows the letter  $i$  in a word.
- $ks_k$  : strength of the finger associated to key  $k$  ( $ks_k \in [0, 1]$ ).
- $d_k$  : distance that the finger attributed to the key  $k$  has to cross to reach that key.

## 2.1 Data's collecting

Concerning the data related to the keyboard ( $sl_k, sr_k, d_k$ ), we have based our analysis on the keyboard present on the BEPO graphic in the assignment paper. The distance is calculated as the number of "moves" from the home row the finger has to do. There is no diagonal moves.

For the strength of the fingers, we have arbitrarily decided that, on a scale of 1, the index will be 0.75, the middle finger 1, the ring finger 0.5 and the little finger 0.1. We have also matched each key with a finger. With the strength of the finger associated, we can thus construct  $ks_k$ .

To collect the frequencies of the appearance of the letters and the bi-grams, we have used the NGramExtraction tool from Deception. To obtain the alphabet that we used, we just take the 47 characters that have the highest number of occurrence (47 is the number of keys in our keyboard). Then we divide the number of occurrence of each letter by the total number of occurrences to have the frequency of each letter. We also "normalize" the bi-gram so that the sum of each bi-gram frequency starting by a specified letter makes always 1.

## 2.2 Data file organisation

The file of data (named `data`) is structured to be also human readable, the first word is the name of the variable, which is followed by its content (note that it has to be in a specified order for the parser program). The data do not have necessarily the same format as in the model, for example for  $sl$  and  $sr$ , it's the number of the key that is indicated. The parser translates it into binary data. We have done this in that way to avoid errors and to keep the data file readable.  $fs$  represent the finger strength and  $kb < x >$  the key associated to a certain finger (there is no key for the thumbs).

## 3 Model

### 3.1 Variables

- $kb_{k,l} = \begin{cases} 1 & \text{if the letter } l \text{ is on the key } k, \\ 0 & \text{otherwise.} \end{cases}$
- $vl = \begin{cases} 1 & \text{if the vowels are on the left side,} \\ 0 & \text{otherwise.} \end{cases}$
- $a_{i,j} = \begin{cases} 1 & \text{if to type } i \text{ and then } j, \text{ it is not the same hand that is used,} \\ 0 & \text{otherwise.} \end{cases}$

### 3.2 Constraints

- $\sum_l fr_l \cdot (\sum_k kb_{k,l} \cdot (sr_k - sl_k)) \geq 0$
- $\sum_k kb_{k,l} = 1, \forall l$
- $\sum_l kb_{k,l} = 1, \forall k$
- $\sum_l vl \cdot (\sum_k kb_{k,l} \cdot sl_k) = V \cdot vl$
- $a_{i,j} = a_{j,i}, \forall i, j$
- $a_{i,i} = 0, \forall i$
- $a_{i,j} \leq \sum_k kb_{k,i} \cdot sl_k + \sum_k kb_{k,j} \cdot sl_k$
- $a_{i,j} \geq \sum_k kb_{k,i} \cdot sl_k - \sum_k kb_{k,j} \cdot sl_k$
- $a_{i,j} \geq \sum_k kb_{k,j} \cdot sl_k - \sum_k kb_{k,i} \cdot sl_k$
- $a_{i,j} \leq 2 - \sum_k kb_{k,i} \cdot sl_k - \sum_k kb_{k,j} \cdot sl_k$

### 3.3 Objective function

$$\min \left[ \sum_l fr_l \cdot \left( \sum_k ks_k \cdot d_k \cdot kb_{k,l} \right) + \sum_i \sum_j w_{i,j} \cdot (1 - a_{i,j}) + \sum_i \sum_j w_{i,j} \cdot \left( \sum_k d_k \cdot kb_{k,i} + \sum_l d_l \cdot kb_{k,j} \right) \right]$$

## 4 Method

To obtain an optimal keyboard related to our model by using the **row generation** method. To implement that method, we use 2 concepts that are present in Gurobi : the lazy constraint and the callback.

The lazy constraints are the constraints that are less likely to be violated, so they are not contained in the set of the beginning but are added as we go along the execution of the solver. In order to do that, we use the callback system. We provide a class containing a `callback()` function which is called at key steps of the resolution. In our implementation, the function is called when the solver found a solution. This function checks if one of the lazy constraints is violated or not. If yes (and is not yet added), we add it to the model by using the `addLazy()` function.

The constraints that are considered lazy are these one :  $\forall i, j$ ,

- $a_{i,j} \leq \sum_k kb_{k,i} \cdot sl_k + \sum_k kb_{k,j} \cdot sl_k$
- $a_{i,j} \geq \sum_k kb_{k,i} \cdot sl_k - \sum_k kb_{k,j} \cdot sl_k$
- $a_{i,j} \geq \sum_k kb_{k,j} \cdot sl_k - \sum_k kb_{k,i} \cdot sl_k$
- $a_{i,j} \leq 2 - \sum_k kb_{k,i} \cdot sl_k - \sum_k kb_{k,j} \cdot sl_k$

It represents a total of  $4 * 47 * 47 = 8836$  constraints.

This method gives us an optimal solution.

## 5 Results

### 5.1 French keyboard

- Sample : "Madame Bovary", Gustave Flaubert.
- Files : `f1-out1.txt` (characters) and `f1-out2.txt` (bigrams)

The result is shown FIGURE 1.

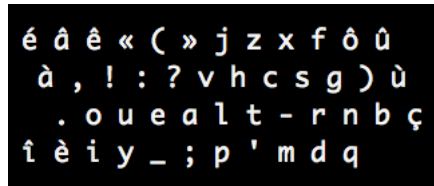


Figure 1: Keyboard : French

There are the numbered results :

Objective function	123.286
Row 1	3.67 %
Row 2	16.89 %
Row 3	60.65 %
Row 4	18.75 %
Time	1024 sec

## 5.2 English keyboard

- Sample : "Pride and prejudice", Jane Austen.
- Files : `pj-1.txt` (characters) and `pj-2.txt` (bigrams)

The result is shown FIGURE 2.

```

< 2 { 0 | " . + z v x #
> u ( } y ) p d r m w [
= a o i e - h l t s , g
1 _ b * ] ; f / c n k

```

Figure 2: Keyboard : English

There are the numbered results :

Objective function	138.185
Row 1	4.19 %
Row 2	17.29 %
Row 3	60.15 %
Row 4	18.14 %
Time	424 sec

## 5.3 Programming keyboard (Language : C)

- Sample : "sched.c", Linux kernel
- Files : `sched-1.txt` (characters) and `sched-2.txt` (bigrams)

The result is shown FIGURE 3.

```

' : & } + 0 q # w v b !
y ( ) . > h - p c k 1 "
o e u i a t r s * n f {
< x _ = / , m ; d l g

```

Figure 3: Keyboard : Language C

There are the numbered results :

Objective function	130.64
Row 1	5.2 %
Row 2	19.28 %
Row 3	56.49 %
Row 4	18.17 %
Time	1456.8 sec

## 6 Improvements

The first improvement that can be done on our model. The (updated) version of our objective function dissociates the difficulty of typing a bigram and if the hands alternates to type that bigram. We could correct this version to link them.

Another possibility of improving the computation is to take the constraints

- $a_{i,j} = a_{j,i}, \forall i, j$
- $a_{i,i} = 0, \forall i$

into account directly in the formulation of the computation in the program, not to give them as "initial" constraint to the solver, this would decrease the number of variables and the number of constraints.