Ergonomic Modelling and Optimization of the Keyboard Arrangement with an Ant Colony Algorithm

Marc Oliver Wagner, Bernard Yannou, Steffen Kehl, Dominique Feillet, Jan Eggers

Ergonomic Modelling and Optimization of the Keyboard Arrangement with an Ant Colony Optimization Algorithm

Marc Oliver WAGNER Bernard YANNOU Steffen KEHL
Dominique FEILLET Jan EGGERS

Corresponding Author:
Bernard YANNOU
Laboratoire Productique Logistique, Ecole Centrale Paris
Grande Voie des Vignes, 92295 Châtenay-Malabry Cédex, France
yannou@pl.ecp.fr
Tel: (33) 1 41 13 16 05, Fax: (33) 1 41 13 12 72

Abstract

The arrangement of letters on a keyboard greatly influences the comfort and the typing speed of the user. With repetitive muscular injuries emerging and the amount of text to be treated increasing, the need for a better arrangement has come forth. In order to solve the problem of this arrangement an abstract representation of a keyboard is introduced. Having justified the choice of heuristic ergonomic criteria, a mathematical function for the ergonomic evaluation criteria is developed. Based on the Ant Colony Optimization algorithm an optimization algorithm is designed and applied to the described problem. The computational results representing the quality of the obtained keyboard arrangements for several languages are finally discussed and new solutions for the keyboard optimization problem are given.

Keywords: keyboard arrangement, product design, ant colony optimization

1 Introduction

A computer user or typist is easily surprised when first looking at the seemingly arbitrary arrangement of letters on a standard computer keyboard. Neither does the arrangement have any alphanumeric logical order nor is it optimized according to a high hit rate or ergonomic comfort. In fact, it is more the result of the historical development of the typewriter market and notably the achieved dominance of the Remington II typewriter which imposed the arrangement. Introduced by the Sholes brothers in 1873, the keyboards using this arrangement have become known as Sholes– or *QWERTY*–keyboards. Initially designed in an English-speaking context, they were slightly changed and thus adapted to other languages like the French and the German. This gave rise to the *AZERTY* and *QWERTZ* versions. These counter–optimized keyboards were justified during some decades during which mechanical problems remained in typewriters. With electronic typewriters and personal computers emerging it was no more the case, but despite some serious work to get optimized keyboard arrangements, the *QWERTY* layout remained the standard.

This fact might be partially explained by pointing out that all new suggestions, notably those issued by August Dvorak in the United States and Claude Marsan in France, have so far been manually designed. This implies a high probability that they are suboptimal and therefore risk to lack the necessary impact to bring about a change of standards. Indeed a repeated change of standards is excluded.

A second obstacle for the introduction of a new standard is the lack of reliable evaluation criteria to justify a change of standards. Ergonomic research in the field of computer keyboards has so far not been able to supply a satisfying set of rules for the ergonomic evaluation of a keyboard arrangement. The only set one can hope for therefore consists of a set of heuristic rules which are derived from common sense and verified by experimental studies.

Using such a set of heuristic criteria based on [16, 14, 15], an evaluation function was established in [12]. The objective of this article is to present the optimization of the arrangement of letters on a keyboard according to those criteria.

The paper is organized as follows: In section 2 the different types of keyboard arrangement problems are presented and a historical overview over the relevant type is given. Section 3 then defines the basic vocabulary, introduces an abstract keyboard model and imposes some restrictions on the general model with respect to the optimization phase. Section 4 is dedicated to the presentation and justification of the heuristic ergonomic rules. In section 5 the procedure developed for obtaining the necessary statistical data is described. In section 6 the algorithm that serves for the optimization procedure is given before section 7 summarizes the computational results in comparing the newly found arrangements to traditional ones. Finally, section 8 concludes the work and gives a perspective for future research.

2 The Keyboard Arrangement Problem (KAP)

2.1 Different Types of Arrangement Problems

Research concerning the arrangement of letters on a keyboard has so far been conducted primarily in two directions and thus has led to two different definitions of a keyboard arrangement problem. The first problem considers the situation where

the number of letters for keyboard exceeds the number of available positions. This leads to an inherent ambiguity, since several letters are assigned to the same key. For instance, these kinds of problem are found for touch-tone telephones where the alphabet consists of 26 letters and the keyboard disposes of only 8 keys and where 2 or 3 letters are assigned to each key (see [18] or the assignment of alphabets based on the Chinese language to European keyboards in [11]). The primary objective is to determine the sets of characters which may be assigned to a same key while causing just a minimum of ambiguity for a given set of words. To cite an example from [18] the sequence "269" on a touch-tone telephone matches the word "boy" as well as the word "box". The number of such words, which have the same representation on the keyboard and which can therefore not be exactly determined from the input, is to be minimized. It is worthwhile to point out, that the problem definition does not consider the question of the association of the character sets to the keys once these character sets have been determined. The set $\{a, b, c\}$ for instance may thus be assigned to any key without changing the result of the optimization problem.

The second type of problem is based on the idea that every character of the alphabet is produced by a unique sequence of keys. Thus, there is no ambiguity in the word representation. The focus of the problem consists of the absolute and relative positioning of the keys on the keyboard in order to optimize a criterion like typing speed [17], rapid learning [17] of the typing system or ergonomics [12].

This paper considers a problem of the second type which will subsequently be called Keyboard Arrangement Problem (KAP). The chosen optimization criteria related to the typing ergonomics are defined in section 4.

2.2 History of the KAP

When the first commercial typewriter was invented in 1867 the notion of ergonomics was practically unknown. The primary criteria for evaluating the difference between the suggested models therefore became the possible hit rate. Due to the mechanical setup of the machines working with little hammers which punched a ribbon in order to produce a letter on the paper, the limiting factor for the hit rate was the inertia of the hammers. When two keys were struck successively, there was the danger of them sticking together and therefore demanding a significant pause to reset the machine. This restriction led to a counter-optimization of the arrangement. Letters were set up to guarantee a maximum distance of the position of consecutive letters and often used letters were positioned at the extremities of the keyboard.

With the invention of the electronic typewriter the mechanical restrictions became obsolete. In the 1930's the American researcher Dr. August Dvorak started his work on a new ergonomic arrangement of the letters allowing a higher typing speed. Due to the lack of computers the research took more than a decade before first results were obtained.

The innovative thoughts were not restricted to the United States. Claude Marsan, a French researcher who had collaborated with Dvorak in the United States slightly modified Dvorak's assumptions on the ergonomics of keyboards and undertook a respective research for the French language.

However, though the American National Standards Institute approved of Dvorak's layout in 1982, neither Dvorak nor Marsan succeeded in imposing their arrangements as a market standard.

3 Modelling of the Keyboard

A keyboard basically serves to enter a string of characters into a memory by striking a sequence of keys. An abstract definition of a keyboard therefore consists of a set of typable characters, the respective sequences of keys which enable their construction and a geometrical setup of the keys. A general keyboard model has to take into account these aspects in order to allow the description of a large family of keyboard, to which the existing standard keyboards inevitably have to belong. In the following section, some important terms to describe the model of the keyboard are introduced.

3.1 Important Definitions

Character A character is the smallest visible unit of a text. It is important to note that "r" and "R" are different characters and that " " and the end of a paragraph are also characters.

Symbol A symbol is the denomination for a character. In most cases the denomination and the character can be identical, however, in some cases a different denomination is useful, e.g. *space* for " ".

Real key A real key is a physical key on the keyboard described by its position.

Special Key A special key is a set of real keys which have the same function and thus the same denomination, e.g. two keys for *shift*. It is defined by its denomination, the set of reals keys it represents and the rule which determines which of the keys is used in a given situation. Special keys need a real key to be struck afterwards in order to produce a character.

Macro A macro is a sequence of real and special keys which activates the character of a given symbol. For all symbol other than space it is unique.

3.2 Definition of a keyboard model

In order to define a keyboard model, four objects have to be defined: The geometric layout, the character and symbol sets as well as the macro list.

Concerning the geometric layout, the fact that keyboards might have a different number of rows, columns and keys has to be taken into account. Most commercial keyboards have a layout which corresponds to a rectangle having rows and columns. Even if those conventional keyboards present offset rows – in a presently unjustified ergonomic way – this is due to ancient mechanical considerations. Each column is generally assigned to a finger and a so called home row is defined where the fingers stay in a relaxed position. Therefore an abstract keyboard layout may be resumed in a key matrix, each key being defined by its geometric position, i.e.:

- a hand (left, right)
- a column (0-7 for the left hand and 0-8 for the right hand, 0 standing for the thumbs)
- a row (0-5, 0 standing for the top row and 3 for the rest row)

The above given number of columns and rows allows the representation of most of the presently available keyboards, notably the *QWERTY* keyboard and its German (*QWERTZ*) and French (*AZERTY*) equivalents as well as the French *Marsan* optimized keyboard [16, 14, 15]. The *AZERTY* keyboard is represented in the abstract model in figure 1.

Subsequently, the optimization is conducted in order to fit on an ergonomic physical keyboard layout, named ECP, with some important patented characteristics [23]. Among those are two palm-rests with integrated keys that can be struck by the thumbs and the forefingers, respectively. The keys for the forefingers are put on an extra row (number 5) while the keys activated by the thumbs are on an extra column (number 0). Authors already worked on fitting the *Marsan* optimized keyboard layout into the ECP layout (see figure 2) leading to the so–called *initial French ECP keyboard*.

As a first alphabet, the complete set of available characters on a standard keyboard is used. The alphabet is subject to simplifications in order to reduce the complexity of the problem (see section 3.3).

For a given keyboard, the macro list is easy to establish, i.e. sequences of keystrokes leading to the symbols. But for a new keyboard to arrange, the sequence of keystrokes leading to a symbol has to be designed in a more or less complex manner. Without any supplementary condition, there is an infinite number of possible layout solutions in terms of macro list solutions, since any sequence of keys may be a candidate for a given symbol. Since a finite solution space is needed to apply one of the metaheuristics designed to solve combinatorial optimization problems, certain restrictions have to be imposed.

3.3 Reduction of the Problem Space

In order to reduce the size of the problem space, certain assumptions have been translated into rules. First of all, the considered alphabet is reduced. Certain characters are not considered because of their rareness in usual texts like "°", "/", *Alt* and *Ctrl*, others like *backspace* because of the impossibility to determine their frequency from a text source. It is worthwhile to notice that the error rate and thus the frequency of *backspace* depends heavily on the arrangement of the keys.

In addition, several keys are imposed to be used in combination in order to reduce the amount of possible macros. Indeed, one adopted the widespread solution for implementing macros as:

- a sequence of one or more *modifier key(s)* like *shift* or *AltGr* followed by one or more *conventional key(s)*
- or a unique *conventional key*

An additional simplification is adopted in fixing position of modifier keys. A useful definition for a future optimization follows:

Combination character set A combination character set is a set of symbols whose macro definitions refer to a same conventional key.

For example, the capital letter "E" is composed by striking *shift* and a key which is used to produce the character "e". Symbols "E" and "e" therefore form a combina-

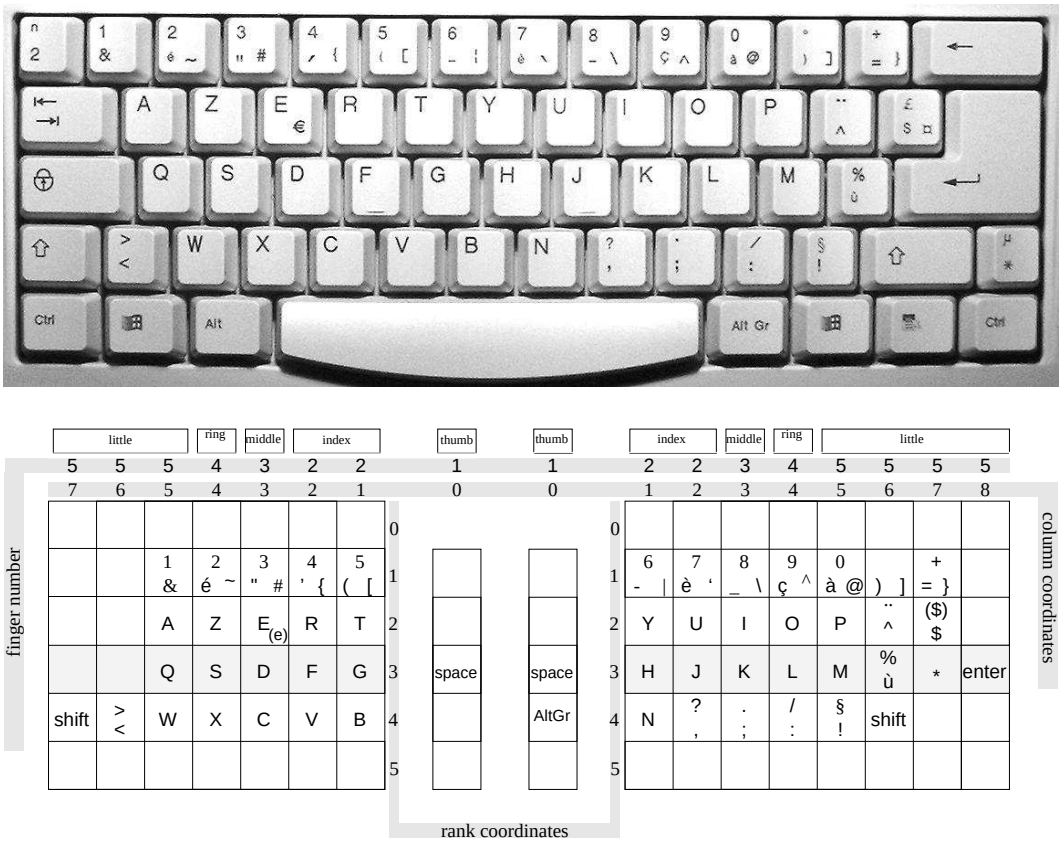

Figure 1: Transcription of the *AZERTY* keyboard

tion character set. The complete combination character set table may be found in [10].

As a consequence, an optimization problem may be considered from the two following notions:

Elementary assignment An elementary assignment is a single association of a combination character set to a real key.

Keyboard solution A keyboard solution is a complete set of elementary assignments, where every combination character set is associated to a real key.

4 Ergonomic criteria

A keyboard design should comply with four main objectives:

- Allow to type a text without fatigue
- Maximize the typing speed
- Reduce the number of typing errors
- Allow a rapid mastery of the touch typing method

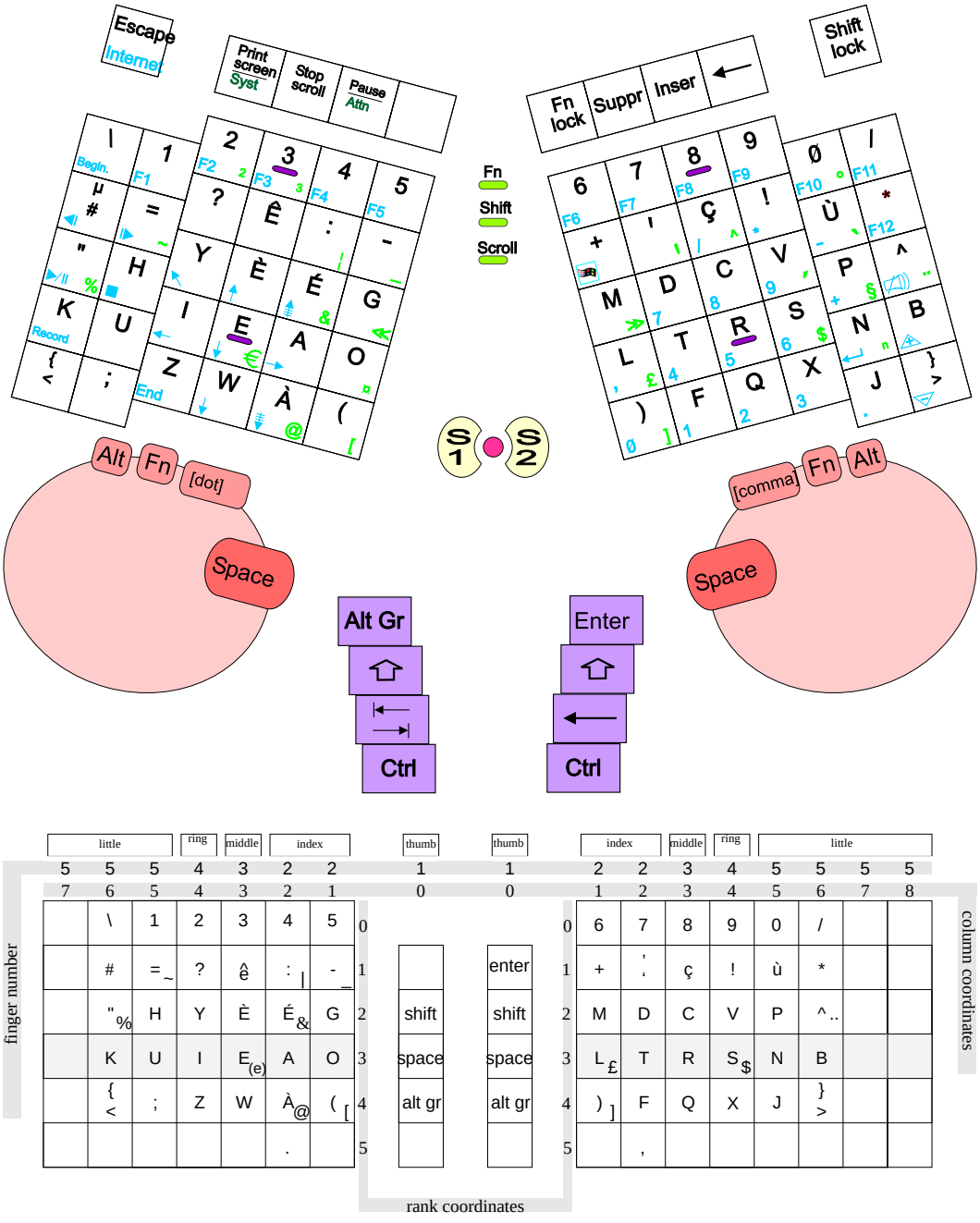

Figure 2: Transcription of the initial French ECP keyboard

As already pointed out, these objectives cannot be easily translated into mathematical criteria which could serve for an evaluation of a given keyboard. Indeed, a first difficulty can be found in the different types of exhaustion and their potential pathological nature. Short–term exhaustion reduces the succeeding typing speed but does not present a danger to the writer's health whereas long term exhaustion might lead to disorders like cumulative trauma disorders. A second uncertainty consists in the localization of exhaustion. Since typing requires movements of fingers, arms and shoulders and since those movements are produced by a combination of actions of muscles, joints and tendons it is difficult to quantify exhaustion. In the following, heuristic criteria are introduced.

4.1 The evaluation function

Because of the lack of exact ergonomic criteria, a set of heuristic rules established in [16], [15] and modified in [12] is used. They are supported by sets of rules established in [17] and [3]. The rules will not be explained in depth nor will an exhaustive justification be given (see [12] for details).

Using a given keyboard layout, any text may be decomposed into a sequence of real keys which have to be struck in order to type the text. This sequence can be seen as a sequence of single real keys – so-called monographs – or a sequence of two real keys – the digraphs. Ergonomic criteria are based upon the distinction between rules that concern the absolute position of a given real key and the relative position of two consecutive real keys.

A first step toward the valuation of a keyboard layout therefore consists of establishing the statistical appearance of mono– and digraphs according to the keyboard used. Subsequently, f_{m_i} and f_{d_i} present the statistical appearance of the monograph m_i and the digraph d_i, respectively. This statistical data serve for the evaluation of the following ergonomic criteria.

Accessibility and Load The combination character sets should be distributed on the keys in a way that the total load is shared in an adequate way by all fingers. The fact that some keys are less accessible than others also has to be taken into account. In order to be able to evaluate a grade, an optimal hit load has to be defined for each key. In a first step, a value is assigned to each hand representing a possible difference between their performances and endurances. Since there is no valid scientific argument for preferring one of the hands, the coefficient is chosen to be 50% for each hand. Then, each column receives a coefficient which takes into account the relative finger agility and endurance. Here, the relative strength of the fingers is known to be greatest for the index finger, then the middle finger followed by the little and the ring finger and finally by the thumb. Their values come from orders of magnitude given by an ergonomist researcher [20]. Then, each row is given a value representing the relative accessibility of that row preferring the home row over all other rows. The optimal load distribution $f_{m_i}^{opt}$ is then calculated by multiplying the three corresponding values for each key. Figure 3 gives a graphical representation of the chosen ideal distribution. It presents the nine columns assigned to the different fingers, starting on the left with one column for the thumb, two columns for the forefinger, a column for the middle and ring finger respectively and four columns for the little finger. The third row represents the home row and the hight of the bars

the ideal hit load normalized to have a maximum of 10000. Additionally, table 1 furnishes the initial values chosen for the rows and the columns.

Nr.	row	column
0	10.87%	15.38%
1	13.04%	10.26%
2	15.22%	15.38%
3	43.48%	23.08%
4	10.87%	17.95%
5	6.52%	6.41%
6		5.13%
7		3.85%
8		2.56%

Table 1: Ideal load distribution

The grade evaluates the variance of the load distribution on a keyboard from the ideal distribution and is given by the following equation

$$v_1 = \sum_{m_i \in \Xi_1^m} (f_{m_i} - f_{m_i}^{opt})^2 \tag{1}$$

where Ξ_1^m is the set of all monographs.

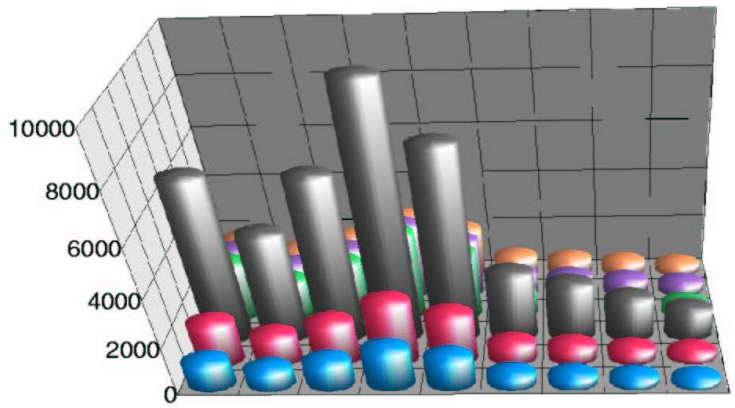

Figure 3: Ideal load distribution for the keys activated by he right hand

Key number In order to minimize the total hit load, the number of hits necessary to compose a given text has to be minimized. The indicator v_2 is therefore calculated by dividing the number of characters present in a text by the number of keystrokes necessary to produce the text. Being relatively important for the most general definition, this grade does not vary between the different solutions of the KAP. This is due to the introduction of the combination character sets which fix the number of keys necessary for each character. It should nevertheless be mentioned that the keyboard which allows the highest productivity in terms of words per minute, the

so–called chord keyboard, requires significantly more hits than a standard keyboard (see [1]).

Hand alternation The fastest and most comfortable typing is assured, when subsequent keys are not hit by the same hand. In order to quantify the hand alternation indicator, we sum the frequency of the digraphs which are typed by using one hand only. In terms of mathematical formulation the indicator is calculated as follows:

$$v_3 = \sum_{d_i \in \Xi_3^d} f_{d_i} \tag{2}$$

where Ξ_3^d is the set of all digraphs which are typed using one hand only.

Consecutive usage of the same finger The preceding alternation rule is true for the fingers as well. Two consecutive keys should not be hit by the same finger. The indicator is calculated by summing the frequencies of the corresponding digraphs and multiplying each of them with a distance coefficient. The greater the distance, the more penalizing a consecutive usage. The relevant set Ξ_4^d is therefore the set of digraphs which are typed using the same finger of one hand. The equation derived is:

$$v_4 = \sum_{d_i \in \Xi_4^d} f_{d_i} dist(d_i) \tag{3}$$

The chosen distance function is the Manhattan distance given by:

$$dist(d_i) = |c_2 - c_1| + |r_2 - r_1| \tag{4}$$

where c_2 and c_1 are the respective columns of the two keys which establish the digraph and r_2 et r_1 the corresponding rows.

Avoid big steps When one hand is used for two consecutive hits, great distances which require awkward hand posture should be avoided. This is the case for keys whose vertical distance is greater or equal to one. The relevant set Ξ_5^d is therefore the set of digraphs which are typed using the same hand, but not the same finger and the vertical distance between the two keys is greater than or equal to one row. A weight coefficient depending on the two fingers used is assigned to each digraph. The coefficient increases with the following pairs of fingers, 1 representing the thumb and 5 the little finger (2-3, 2-5, 3-5, 2-4, 3-4, 4-5). Finally, the indicator is given by:

$$v_5 = \sum_{d_i \in \Xi_5^d} \kappa(d_i) f_{d_i} \tag{5}$$

where $\kappa(d_i)$ is the weight coefficient. The chosen heuristic values for $\kappa(d_i) = \kappa(i, j)$ are represented in table 2, i and j representing the first and second finger respectively.

	thumb	forefinger	middle finger	ring finger	little finger
thumb	0	0	0	0	0
forefinger	0	0	5	8	6
middle finger	0	5	0	9	7
ring finger	0	8	9	0	10
little finger	0	6	7	10	0

Table 2: Big steps coefficients

Hit direction For digraphs typed by using one hand only the preferred hit direction is from the little finger towards the thumb. This is the natural finger movement for most of people, which may easily be verified by tapping on the table according to the two possible directions. Ξ_6^d is therefore the set of all digraphs which are produced by using one hand only and whose hit direction is not the preferred one. The indicator is given by:

$$v_6 = \sum_{d_i \in \Xi_6^d} f_{d_i} \tag{6}$$

Global grade In order to define a global grade, let us consider the six indicators v_j $(1 \leq j \leq 6)$. Since these indicators have different ranges and units and since they have different relative importance, they are divided by the respective indicators of a reference keyboard $v_{j,ref}$. Once the indicators are turned dimensionless they can be multiplied by a relative weight coefficient γ_j and summed. This relative global grade which will be our optimization criterion expresses the fact that one will try to improve at best keyboards from existing keyboards (in practice, they will be the English *QWERTY*, the French *AZERTY* and the German *QWERTZ* keyboards).

The values of weights γ_j are represented in table 3.

Rule	γ
load and accessibility	0.45
Key number	0.5
Hand alternation	1.0
Consecutive usage of the same finger	0.8
Avoid big steps	0.7
Hit direction	0.6

Table 3: Weight coefficients γ

This leads to the final formula:

$$V = \sum_{j=1}^{6} \frac{v_j}{v_{j,ref}} \gamma_j \tag{7}$$

that defines the evaluation of a keyboard for the KAP. The lower the grade is, the better is the keyboard arrangement.

5 Statistical Data

In order to evaluate the subsequent ergonomic criteria, two lists must inevitably be established for a given language: The list of monographs of keys and the list of digraphs of keys. These two lists are generated by parsing a given text with the macro definitions and thus transcribing the sequence of characters into a sequence of real keys. It is three major newspapers/magazines that serve as the source for the text which is parsed: *USA Today* for the English language, *Le Monde* for the French and *Der Spiegel* for the German.

First of all, each character has to be assigned to a symbol in order to give each symbol a name tag. The most natural way is to assign to each character the character itself as a symbol and to name functional keys according to their function, e.g. *shift* and *return*.

To give an example of the procedure, the parsing of the sequence "le Ce" which is part of the string "Ecole Centrale" – the name of the institution where the presented research has been conducted – is demonstrated in the following.

As a second source of information, the definition of the virtual and real macros has to be known.

5.1 Virtual Macro List

The virtual macro list serves as an intermediate stage between the physical model of the keyboard and the real macro list. Indeed, at this stage special keys involved in the virtual macro list are not instantiated into real keys (see next section). In table 4 an extract of the virtual macro list, corresponding to the string "le Ce" is given for the *AZERTY* keyboard - keys are described as (hand, column, row).

symbol	key 1	key 2	symbol	key 1	key 2
c	(0,3,4)		C	Sh	(0,3,4)
e	(0,3,2)		E	Sh	(0,3,2)
l	(1,4,3)		L	Sh	(1,4,3)
lSp	(0,3,0)		rSp	(1,3,0)	
lSh	(0,7,4)		rSh	(1,6,4)	

Table 4: Partial list of virtual macros

5.2 Decision rules

For all special keys like *shift*, *AltGr* etc. a decision rule has to be defined which determines which of the real keys that are part of the special key has to be struck. On the physical model, the special key has to be pressed while the real key that follows is struck. Since releasing a key does not require a significant amount of concentration or time, only the striking is taken into account. According to standard typing rules, both hands should be used to produce the character. The decision rule can therefore be stated to be: The hand which strikes the following key is determined, the special key being struck by the other hand.

The only special macro which is not unique is the one for *space* since the key might be struck by any hand. In order to determine the hand which strikes the key, the following rule is used: The hand which strikes the previous key is determined and *space* is struck by the other hand.

These two rules being defined, the virtual macro list can be transcribed into a real one.

5.3 Real macro list

When the symbols are replaced by real keys according to the corresponding rules explained in 5.2, the virtual macro list is transformed into the real one. In the given example, *space* has to be replaced by *rspace*, since the preceding key is struck by the left hand and then substituted by the position of the physical key. This leads to the list given in table 5. All the necessary elements being defined, the two major lists can finally be established.

symbol	key 1	key 2	symbol	key 1	key 2
c	(0,3,4)		C	(1,6,4)	(0,3,4)
e	(0,3,2)		E	(1,6,4)	(0,3,2)
l	(1,4,3)		L	(0,7,4)	(1,4,3)
lSp	(0,3,0)		rSp	(1,3,0)	
lSh	(0,7,4)		rSh	(1,6,4)	

Table 5: Partial list of real macros

5.4 List of monographs of keys

In order to obtain the sequence of keys, the sequence of characters has to run through the following substitutions: Each character is first replaced by a symbol which represents the character. Then, the symbol is substituted by the corresponding special macro, which is unique for each symbol. Finally the special macro is transformed into a real macro by taking into account the decision rules. The whole process is illustrated in figure 4. As the final result, the sequence of real keys is obtained and

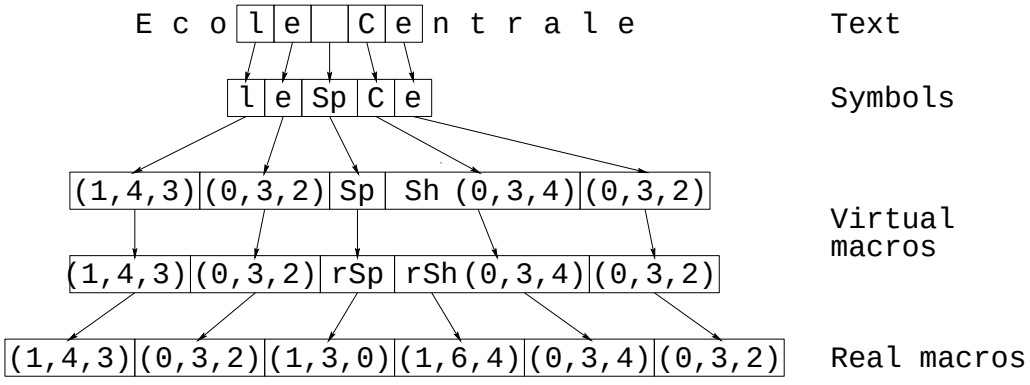

Figure 4: Translation of a source text into a series of keystrokes

the frequency of each key that occurs in the text calculated by dividing its number of occurrences by the total number of occurrences of monographs (see table 6). The list of the digraphs of keys is obtained in an analogous way.

One can notice that our apparently complicated model allows a discoupling between the frequencies of characters found in a text and those of physical key strokes. This is not the case for most of the previously conducted studies of the keyboard arrangement. Moreover, it allows to take into account the difference between upper–case and lower–case letters.

key	frequency	key	frequency
(1,4,3)	0,166	(0,3,2)	0,333
(1,3,0)	0,166	(1,6,4)	0,166
(0,3,4)	0,166		

Table 6: List of monographs

5.5 Final realization of the data model

Since the text sources are of extensive length (greater than 10 Mb), a data model is established which does not directly parse the source text but which first establishes a statistic for the mono-, di- and trigraphs of characters which appear in the text. Subsequently, the parsing of these data suffices in order to establish the lists of mono- and digraphs of keys. A detailed description of the working of the corresponding parsing algorithm can be found in [12]. The data flow from the four initial objects (source text, physical keyboard, evaluation coefficients and the dictionary) to the final statistical results is presented in figure 5.

6 Implementation of the optimization algorithm

The KAP is a discrete, combinational optimization problem. The evaluation function is neither linear nor convex. As such, the natural solution strategy may be found in metaheuristic optimization algorithms such as taboo search, genetic algorithms, simulated annealing or Ant Colony Optimization (ACO) algorithms. Another important characteristic of the evaluation function should be stated: Compared to more classical problems as the Travelling Salesman Problem (TSP) or the Quadratic Assignment Problem (QAP), the time necessary to calculate the global grade is excessive. An efficient algorithm should therefore not demand the global evaluation of many solutions.

Most optimization algorithms like simulated annealing and genetic algorithm and most local search procedures use the global grade and an abstract memory as the only means to direct the search. In contrast, ACO algorithms additionally take advantage of local information in order to construct solutions as described in the following. This local information and the memory are both easily accessible and relatively cheap in terms of computational time. The successful application of ACO algorithms to several different optimization problems like the TSP and the QAP

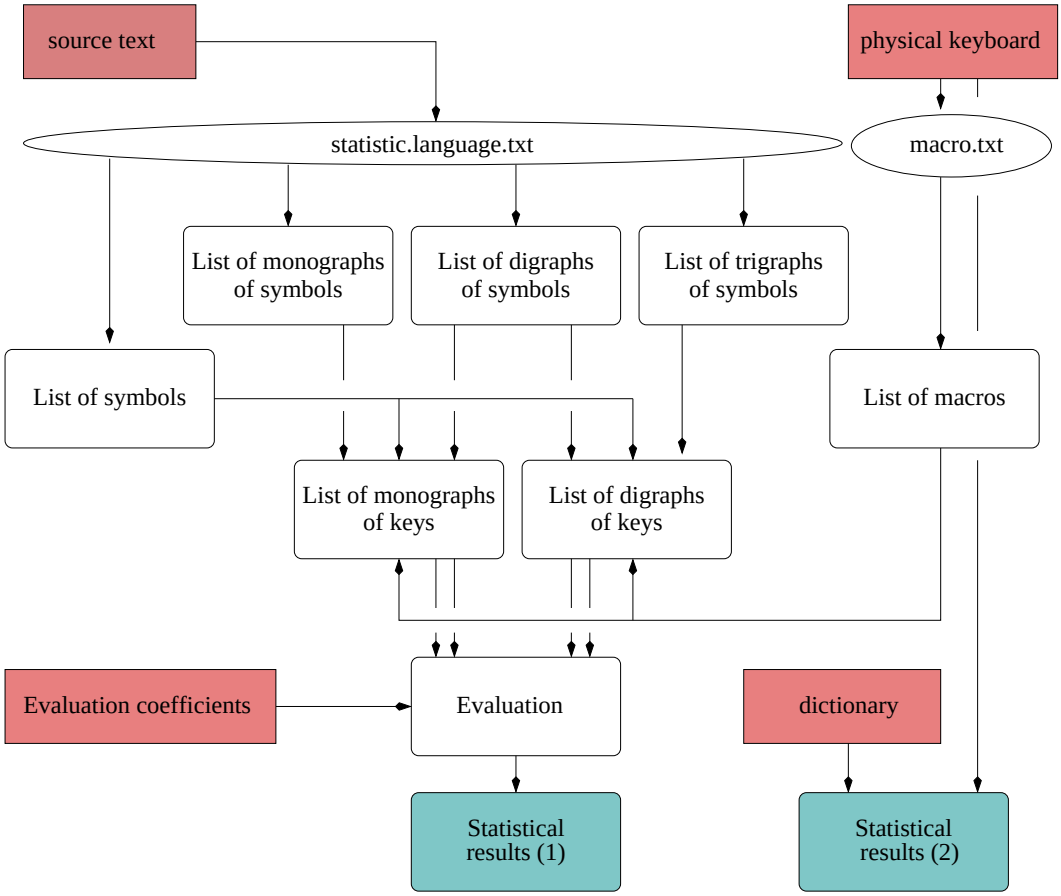

Figure 5: Data flow

finally led to the conclusion that their potential to solve the KAP might exceed the potential of other approaches.

6.1 An ACO based algorithm for the KAP

Ant Colony algorithms were first introduced for solving combinatorial problems in [7] and have subsequently been applied to several other optimization problems. So far, they have proven to be a powerful and easily adaptable tool which yields excellent results over highly combinatorial problems. For these reasons, they were chosen as the metaheuristic framework for the optimization procedure. The fundamental idea of ACO algorithms is based on the behavior of natural ants who succeed in finding shortest paths from their nest to food sources by communicating via a collective memory which consists of pheromone trails. Due to its weak global perception of its environment, an ant moves essentially at random when no pheromone is present. However, it tends to follow a path with high pheromone level when many ants move in a common area, which leads to an autocatalytic process. Finally, the ant does not choose its direction based exclusively on the level of pheromone, but also takes into account the proximity of the nest and the food source respectively. This allows the discovery of new and potentially shorter paths.

Applied to the keyboard assignment problem, the algorithm can be defined as follows: A colony consists of N ants. Each ant is given a string of characters, that is extracted from the reference text which serves as the source for the statistical data, and an empty keyboard, i.e. a keyboard without a macro list. It then arranges the combination letter sets that correspond to the characters of the string on its own keyboard in a random manner, but taking into account successful placements in the past and respecting as much as possible the ergonomic rules established in 4. Once every ant has assigned every combination character sets to a key, a cycle is finished and the keyboards are evaluated. The ants place pheromone on the elementary assignments which are part of its solution. These pheromone placements are summarized in a pheromone matrix (τ_{ij}) where i is a combination character set and j the associated key. The amount of pheromone that each ant places is proportional to the quality of their solution. Then the pheromone is partially evaporated, the keyboards are cleared and the ants restart the search process with new character strings until a predefined stopping criterion is met.

6.2 Principles of the algorithm

The algorithm is a combination of several versions suggested by the ACO research community. As a basic framework, the ACO algorithm presented in [6] was used since it displays the most promising features. By emphasizing the importance of the best solution found, the basic ideas of AS_{Elite} [7, 8, 6] are added by assigning more pheromone to the ant that found the best solution during one cycle. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant Systems [6, 22] contribute to the final algorithm, because a minimum and maximum are defined for the pheromone level. Finally, AS_{Rank} [2] introduces the choice of the ants that are allowed to update the pheromone according to the respective ranking of their solutions.

Since there are some interesting starting points for the search in the form of existing good quality keyboard arrangements like the *Dvorak* or the *Marsan* keyboards, it is possible to modify the standard ACO algorithm which demands a uniform initial distribution of pheromone and to consider a non–uniform initialization of the pheromone–matrix. Such a distribution – first suggested in [4] – translates an a–priori knowledge of the subspace in which the optimal solution may be found.

The obvious inconvenience of this initialization is the fact that an optimal solution outside the determined subspace is found with a smaller probability than in the case of a uniform initial pheromone distribution.

A second modification is due to the fact that it is intractable to guarantee that all the characters are present in the text string used by a given ant to establish its keyboard. Indeed, some rarely used combination character sets are usually not attributed during a cycle. This possibly results in an incomplete keyboard at the end of the construction phase and prohibits a valid evaluation of the quality of the keyboard. Due to their minor importance, the remaining sets are randomly assigned to the remaining keys once the end of the character string in reached.

6.3 Description of the algorithm

From the previously mentioned principles, the following algorithm has been deduced. The algorithm consists of two parts. The first one is an initialization phase which

creates the necessary data structure and sets all parameters to their initial value. The second part is the iterative phase of the algorithm which is repeated until a stopping criterion is satisfied. This criterion is defined to be a certain amount of cycles.

6.3.1 Initialization

During the initialization phase, the following tasks are carried out:

1. An Ant Colony of N agents is created.
2. A text source is defined.
3. An empty character string of fixed length l is assigned to each ant.
4. An empty keyboard is assigned to each ant.
5. A pheromone matrix is created and its elements are set to values that take into account an a–priori knowledge of the solution in the form of high quality keyboard arrangements.
6. The algorithm parameters (see definitions later) are initialized.

6.3.2 Iteration

The second part of the algorithm is repeated until a certain number of iterations has been completed. The subsequent steps are:

1. The keyboards assigned to the ants are emptied.
2. The strings assigned to the ants are initialized using the text source.
3. Each ant reads its string and places the combination character sets on its keyboard until the end of the string is reached. This procedure is described in detail in section 6.3.3.
4. The remaining combination character sets are randomly assigned to the remaining keys.
5. Each ant evaluates the solution it has found.
6. The pheromone matrix τ evaporates with a coefficient ρ_{all} :

$$\tau \leftarrow \rho_{all} \cdot \tau \tag{8}$$

7. A number p of the best solutions are selected and pheromone is updated according to their rank: The k^{th} best ant, with $1 \leq k \leq p$, deposits a quantity $\Delta\tau \cdot q(k)$ of pheromone on each elementary assignment that it has chosen, where $\Delta\tau$ represents a quantity of pheromone and the vector q is a measure of importance of the best ants.
8. The pheromone matrix is modified to fit the allowed range $[\tau_{min}, \tau_{max}]$.

A representation of the algorithm in pseudocode is given in figure 6.

```
Choose the number N of ants;
Set Nt := number of keys;
Set Nc := number of combination character sets;
Set l:= length of a text string;
Initialize the matrix of pheromones P[Nt][Nc];
Repeat
  Create N text strings S[N][l];
  Create N empty keyboards sets T[N][Nt];
  For i := 1 to l do
    For k := 1 to N do
      c := S[k][i];
      If letter c was not already treated by ant k then
        Choose a position p for c;
        T[k][p] := c;
        Evaporate P[p][c];
      EndIf;
    EndFor;
  EndFor;
  For all ants k := 1 to N do
    Evaluate the result for the ant k;
  EndFor;
  Choose the best results;
  Update P[][];
  Verify Min-Max of P[][], modify if necessary;
Until satisfying result;
```

Figure 6: Solution algorithm for the KAP

6.3.3 Character placement

The subsequent steps allow an ant to choose the most adequate key to place the combination character set i under consideration:

1. For a given character to process, the ant verifies if its corresponding combination character set has already been placed on the keyboard. If so, the ant skips the following steps and continues with the next character of the string.

2. For all free keys j, the ant calculates a probability according to the random–proportional decision rule given in equation 9:

$$p_{ij} = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{k \in \Omega} [\tau_{ik}]^{\alpha} \cdot [\eta_{ik}]^{\beta}} \tag{9}$$

   where Ω is the set of not yet occupied keys and η_{ij} the global grade calculated according to the formula given in section 4.1 when only the present and the previously assigned keys are considered. It is worthwhile to note that the grade η_{ij} only needs a few calculations to be computed. α and β are two coefficients describing the relative importance of experience and locally good elementary assignment. The ant uses the calculated probabilities to establish a random variable, according to which it determines a key among those that are still free.

3. The ant places the combination character set on the chosen key.

4. The ant evaporates pheromone on the elementary assignment found according to the following equation:

$$\tau_{ij} \leftarrow \rho \cdot \tau_{ij} \tag{10}$$

ρ being the coefficient of evaporation.

More details concerning ACO approaches of the literature and our specific ACO adaptation for the keyboard design are given in [9].

7 Computational Results

The experimental work consisted of two major phases. First, an efficient parameter setting was determined. Then, the algorithm was run several times using this setting with and without initializing the pheromone matrix.

7.1 Parameters setting

It is known that the convergence depends strongly on the parameters affecting the computational formula (see [7]). In the case of the presented algorithm, N, α, β, ρ, ρ_{all}, $\Delta\tau$, τ_{min}, τ_{max}, p, l and q had to be determined. As a starting point for a heuristic determination of the parameters, those used in [7] were chosen. Observing the development of the pheromone matrix, the parameter settings were adjusted experimentally. The best parameters found are summarized in table 7.

N	30	α	3.0	β	3.0
ρ	0.98	ρ_{all}	0.95	$\Delta\tau$	0.2
τ_{min}	0.09	τ_{max}	1.0	p	4
l	6000	q	$[1, 0.5, 0.2, 0.1]$		

Table 7: Choice of optimization parameters

Since the objective consists of finding a nearly optimal keyboard arrangement, the algorithm is not meant to be for repetitive use. Once a quasi-optimal solution is found, the algorithm is not used any more, the only apparent reuse being the application of the algorithm to other languages. In order not to waste too much time, a statistical examination of different parameter settings was left out. However, slight changes in some parameters frequently led to a significant decrease of the solution quality.

7.2 Optimized keyboard arrangements

As a direct output of the Ant Colony Optimization algorithm, the best arrangement found for the French language is presented in figure 7. It is 51% better than the French *AZERTY* standard. It needed about 2000 iterations which corresponds to about 14 hours on a 400 MHz PC. The best arrangement found for the English language is given in figure 8 and is 41% better than the English *QWERTY* standard.

The best arrangement found for the German language is given in figure 9 and is also 41% better than the German *QWERTZ* standard.

These arrangements require six rows and therefore can not be mapped on the standard keyboard layout ; they require an ergonomic ECP layout (described in [23]) as presented in figure 2.

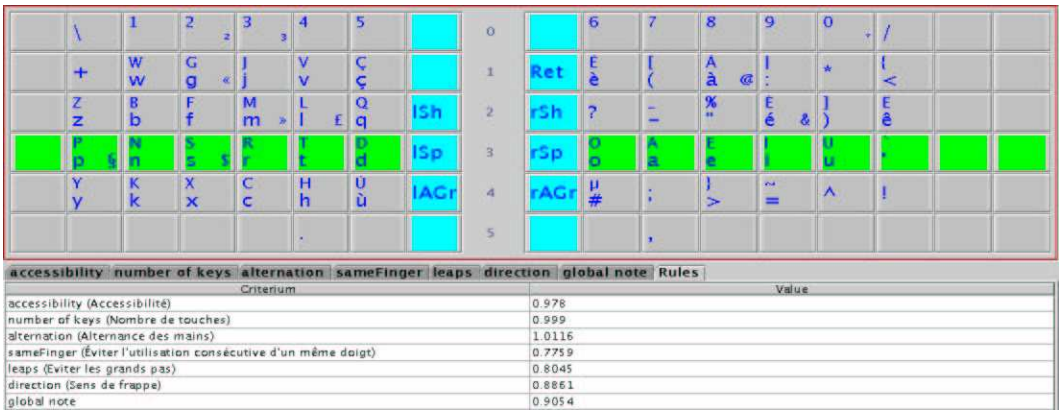

Figure 7: The algorithm output for the French language, evaluated in comparison to the initial ECP arrangement

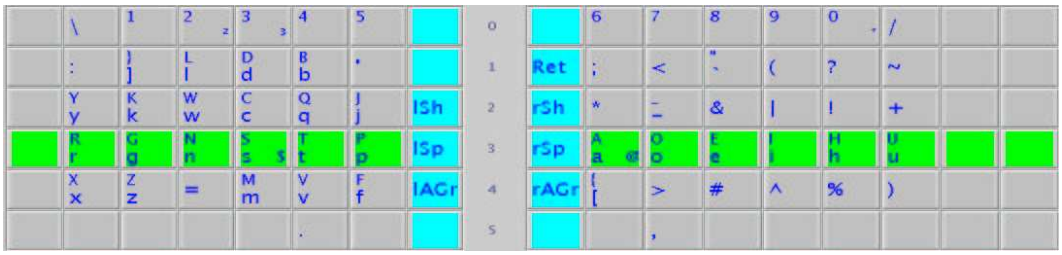

Figure 8: The algorithm output for the English language

7.3 Quality of designs

The best keyboards produced by the algorithm always have the same main characteristics. They have in common – though they were not identical – that all the vowels are struck by the same hand and that no frequently used consonant is found on that side of the keyboard. The relative position of the vowels and the most frequently used consonants are identical for most of the runs. This partitioning property is similar to those of *Dvorak* (in the English context) and *Marsan* (in the French context) manually optimized keyboards, which is intellectually satisfactory. Far more, the found keyboards outrank these well known *Dvorak* and *Marsan* manually optimized keyboards. The keyboard qualities have been summarized in table 8.

For the French language, the optimization algorithm has improved by 19% the *Marsan* keyboard. Authors previously worked on a manually adaptation of the

Figure 9: The algorithm output for the German language

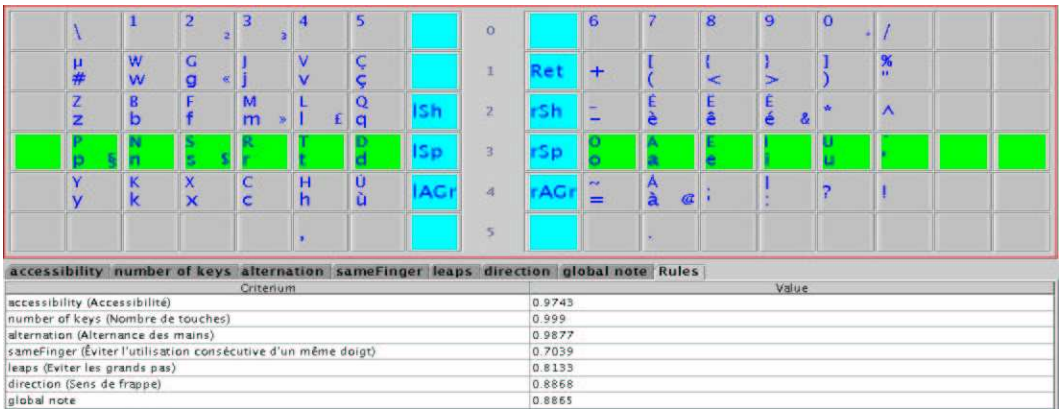

Figure 10: The final ECP arrangement for the French language, evaluated in comparison to the initial ECP arrangement

Marsan arrangement [16, 14, 15] over the ECP physical layout, leading to an *initial ECP arrangement* keyboard given in figure 2. This keyboard was already 11.5% better than the *Marsan* keyboard but the optimization algorithm found a keyboard design (see figure 7) which is 9.4% better than this initial ECP arrangement. Finally, a manual additional modification was performed in order to a reinforce certain symmetries and proximities which were not considered in the ergonomic criteria. It leads to our definitive optimized proposition, named final ECP arrangement, represented in figure 10. Reinforcing certain symmetries and proximities facilitates memorizing the new positions of the letters on the keyboard. For example, brackets are positioned symmetrically and certain characters are grouped together, like “e”, “é”, “è” and “ê”. These modifications, that were partially not allowed to be affected by the algorithm, further slightly increase by 1.9% the quality of the keyboard arrangement. This ultimate arrangement has finally been suggested as a standard layout for future French keyboards.

These good design quality improvements might partially be explained by the lack of adequate computational machines in the time that the majority of work in the area of the redistribution of the letter was realized. Therefore, most previous work in the domain led to simplifications in the models which introduced several minor disadvantages. The fact that *space* and the difference between ordinary and capital letters was neglected is just one example. However, certain aspects found during the optimization match the simplifications which prior researchers assumed

	AZERTY	*Marsan*	initial ECP arrang.	algo. output	final ECP arrang.
load and accessibility	1	0.489	0.491	0.479	0.478
hand alternation	1	0.687	0.665	0.673	0.657
consecutive usage of the same finger	1	0.503	0.338	0.263	0.238
avoid big steps	1	0.315	0.276	0.222	0.224
hit direction	1	0.454	0.421	0.373	0.373
global grade	1	0.568	0.518	0.487	0.480
improvement vs *AZERTY*	0%	43.2%	48.2%	51.3%	52.0%
improvement vs *Marsan*		0%	11.5%	17.9%	19.3%
improvement vs initial ECP			0%	9.4%	11.3%

Table 8: Comparisons between the French keyboards: The standard *AZERTY*, the manually optimized *Marsan*, the initial ECP arrangement, the ant colony algorithm output, the final ECP arrangement

to hold. Just to cite one example, Marsan established in [16, 14, 15] a rule that demanded that all vowels be put on the same side of the keyboard. Though not demanding such a rule, all algorithm outputs had in common that this rule was fulfilled.

For the English language, the optimization algorithm has improved by only 1.9% the *Dvorak* keyboard (see figure 8). This apparently disappointing result proves that the English keyboard was more studied by researchers. But our program tends to show at least that the *Dvorak* keyboard arrangement is close to an optimal keyboard design.

7.4 Illustrative Criteria

In order to compare the solution keyboard to the standard ones in a more illustrative way, a second, more illustrative set of evaluation criteria has been established.

Mean Distance travelled by a Finger In order to strike a key the concerned finger leaves the relaxed position on the home row, moves to the designated position, strikes the key and returns to the initial position. In order to evaluate the average distance travelled during one keystroke, the size of a key is assumed to be $1,9{*}1,9\text{cm}^2$ and the keys are assumed to be arranged on a rectangular scheme. For the *AZERTY* keyboard this method is advantageous, since the real distance travelled is greater due to the fact that the rows are shifted against each other. The indicator is calculated according to the following formula:

$$dist = \sum_{m_i \in \Xi_1^m} 2 \cdot f_{m_i} \gamma(m_i) \tag{11}$$

where Ξ_1^m is the set of all monographs and where γ is the Euclidean distance between the position of the key on the home row where the finger rests and the key struck.

The following criteria are the absolute values for some of the previously presented relative indicators. They are all calculated by summing the frequencies of the concerned mono– and digraphs, respectively.

Right hand usage For a load equilibrium between the right and the left hand, the percentage of monographs that are struck by the right hand might be considered.

Hand alternation This indicator gives the percentage of digraphs that are struck by using both hands and thus respecting the corresponding, previously stated rule.

Disadvantageous direction When using one hand, typing in the direction from the thumb toward the little finger is disadvantageous.

Big and small steps The percentage of digraph struck by a single finger which must travel one row or more than one row is given by the indicator *small* and *big*, respectively.

The results *AZERTY* keyboard, the *Marsan* keyboard and the final ECP arrangement keyboard are compared in table 9 in terms of these illustrative criteria.

	AZERTY	*Marsan*	final ECP arrangement
mean distance	3.07cm	1.69cm	1.48cm
right hand usage	45.8%	52.5%	52.9%
hand alternation	60.9%	73.1%	74.3%
disadvantageous direction	13.1%	6.0%	4.6%
big steps	2.11%	0.72%	0.25%
small steps	2.54%	1.22%	1.27%

Table 9: Illustrative criteria for *AZERTY*, the *Marsan* and the final ECP arrangement keyboards

8 Conclusions

In this paper, the ergonomic optimization of the distribution of the characters over an ergonomic keyboard layout – described in [23] and represented in figure 2 – has been realized for the French, the English and the German languages. In order to do so, a flexible keyboard model has been introduced and coupled with a statistical analysis of the concerned languages. Second, six ergonomic criteria have been mathematically defined. Third, an *Ant Colony Optimization* algorithm has been proposed which is based on several different ant optimization approaches found in the literature. The obtained results show that the algorithm is effective in finding very good solutions. In particular, all known manually optimized keyboards such as the *Dvorak* keyboard and the *Marsan* keyboard were outranked. This gain in quality might be explained by pointing out that previous researches primarily focused on typing speed or on easy memorization of the arrangement. The new results concern a comprehensive approach of the ergonomic quality which the mentioned criteria are just one part of. All the described modelling and optimization process can be applied to any other language as soon as statistical data are established from a representative text.

As the reality of typewriting can not be completely taken into account in a mathematical function, these promising results obviously need a practical verification.

References

[1] David G. Alden, Richard W. Daniels, and Arnold F. Kanarick. Keyboard design and operation: A review of the major issues. *Human Factors*, 14(4):275–293, 1972.

[2] Bernd Bullnheimer, Richard F. Hartl, and Christine Strauß. A new rank based version of the ant system – a computational study. Technical Report POM-10/97, Institute of Management Science, University of Vienna, April 1997.

[3] R.E. Burkard and J. Offermann. Entwurf von Schreibmaschinentastaturen mittels quadratischer Zuordnungsprobleme. *Zeitschrift für Operations Research*, 21:B121–B132, 1977.

[4] D. Corne, M. Dorigo, and F. Glover. *New Ideas in Optimization*. McGraw-Hill, 1999.

[5] M. Dorigo and L.M. Gambardella. Ant colony systems: A cooperative learning approach to the travelling salesman problem. *IEEE Transactions of Evolutionary Computation*, 1:53–66, 1997.

[6] Marco Dorigo, Gianni Di Caro, and Luca M. Gambardella. Ant algorithms for discrete optimization. Technical report, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium and IDSIA, Lugano, Switzerland.

[7] Marco Dorigo, V. Maniezzo, and A. Colorni. Ant system: An autocatalytic optimizing process. Technical Report 91-016, Dipartimento di Elettronica e Informazione, Politechnico di Milano, Piaza Leonardo da Vinci 32, 20133 Milano, Italy, 1991.

[8] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transaction on Systems, Man and Cybernetics - Part B*, 26(1):1–13, 1996.

[9] J. Eggers, D. Feillet, S. Kehl, M.O. Wagner, and B. Yannou. An ant colony optimization algorithm for the optimization of the keyboard arrangement problem. Technical Report CER 01-03 A, Laboratoire Productique Logistique, Ecole Centrale Paris, august 2001.

[10] Jan Eggers, Steffen Kehl, and Marc Oliver Wagner. élaboration des algorithmes d'optimation d'un clavier. Applied scientific project, Laboratoire Production et Logistique, École Centrale Paris, august 2001.

[11] David Euge Glover. *Experimentation with an adaptive search strategy for solving a keyboard design/configuration problem*. PhD thesis, University of Iowa, 1986.

[12] Steffen Kehl and Marc Oliver Wagner. Modélisation et optimisation de la répartition des lettres sur un clavier d'ordinateur. Applied scientific project, Laboratoire Production et Logistique, École Centrale Paris, july 2000.

[13] Vittorio Maniezzo and Antonella Carbonaro. Ant colony optimization: An overview. Technical report, Scienze dell'Informazione, University of Bologna, Italy, june 1999.

[14] Claude Marsan. Demande de certificat d'addition no. 79-01065, du 17 janvier 1979 portant sur le brevet no.76-23323 : "Perfectionnements aux claviers de machines á écrire et similaires", Institut National de la Propriété Industrielle, France.

[15] Claude Marsan. Claviers alphanumériques ergonomiques pour machines à écrire et similaires. brevet d'invention no. 87-03267, déposé le 3 mars 1987 à l'Institut National de la Propriété Industrielle, France.

[16] Claude Marsan. "Perfectionnements aux claviers de machines à écrire et similaires". brevet d'invention no. 76-23323, déposé le 30 juillet 1976 à l'Institut National de la Propriété Industrielle, France.

[17] Donald A. Norman and Diane Fisher. Why alphabetic keyboards are not easy to use: Keyboard layout doesn't much matter. *Human Factors*, 24(5):509–519, 1982.

[18] B. J. Oommen, R.S. Valiveti, and J. R. Zgierski. Correction to "an adaptive learning solution to the keyboard optimization problem". *IEEE Transactions on Systems, Man and Cybernetics*, 22(5):1233 – 1243, 1992.

[19] D. T. Pham and Dervis Karaboga. *Intelligent Optimisation Techniques*. Springer-Verlag, mai 2000. ISBN: 1852330287.

[20] James Richardson, 2000. Private communication.

[21] Thomas Stützle and Marco Dorigo. ACO algorithms for the traveling salesman problem. Technical report, IRIDIA, Université de Bruxelles, Belgium, 1999.

[22] Thomas Stützle and Holger H. Hoos. Improving the ant system: A detailed report of the MAX–MIN ant system. Technical report, Technical University of Darmstadt, Department of Computer Science – Intellectics Group, Alexanderstr. 10, 64283 Darmstadt, Germany.

[23] Bernard Yannou and Paul Hossenlopp. Clavier alphanumérique ergonomique. brevet d'invention no 98-04588, déposé le 10 avril 1998, délivré le 23 juin 2000 sous le numéro de publication 2 777 222.

List of Figures

List of Tables