An Ant Colony Optimization Algorithm for the Optimization of the Keyboard Arrangement Problem

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

An Ant Colony Optimization Algorithm for the Optimization of the Keyboard Arrangement Problem

Jan EGGERS Dominique FEILLET Steffen KEHL
Marc Oliver WAGNER Bernard YANNOU

Laboratoire Productique Logistique, Ecole Centrale Paris
Grande Voie des Vignes, 92295 Châtenay-Malabry Cédex, France
feillet@pl.ecp.fr, yannou@pl.ecp.fr

Abstract

In order to solve the problem of the arrangement of letters on a computer keyboard, an abstract representation of a keyboard is introduced and an evaluation function taking account of ergonomic criteria is proposed. Based on the generic framework of Ant Colony Optimization, an algorithm is developed and applied to the described problem. New effective keyboard arrangements are deduced for several languages. Comparisons are made with standard manually optimized keyboards.

Keywords: evolutionary computations, ant colony optimization, keyboard arrangement

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. The keyboard was even optimized to slow down typists to prevent the keys getting stuck. 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 Anglophone context, they were slightly changed and thus adapted to other languages like the French and the German. These

counter–optimized keyboards were justified during some decades whereas mechanical problems remained in typewriters. Rapidly it was no more the case but despite some serious work to get optimized keyboard arrangements (*Dvorak* in the USA and *Marsan* in France [18]), the *QWERTY* layout remained the standard.

This fact might be partially explained by the drawbacks to a standard change, which excludes repeated changes or changes with no guarantee of a prominent impact. As a matter of fact, 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 collection of heuristic rules which are derived from common sense and verified by experimental studies.

Some attempts have been performed to propose a keyboard arrangement optimizing a criterion like typing speed or rapid learning of the typing system [20] but, so far as we know, no paper has been devoted to the optimization of a keyboard arrangement taking account of more precise ergonomic criteria.

The aim of this paper is to describe the application of an Ant Colony Optimization (ACO) algorithm to the problem of the arrangement of letters on a computer keyboard (KAP, for Keyboard Arrangement Problem), using an evaluation function based on a complex set of ergonomic criteria. Let us mention that a complementary paper [25], designed from an ergonomic facet, focuses more heavily on the definition of ergonomic criteria, their respective importance and the analysis of the results.

Up to now, ACO algorithms have already been used to solve very different types of optimization problems. Among these problems are the Traveling Salesman Problem (TSP) [8, 10, 22], the Quadratic Assignment Problem (QAP), the sequential ordering problem [12], the graph coloring problem [7] and several other combinatorial, static optimization problems as well as dynamic ones which are primarily found in the routing for telecommunication networks [4, 5]. ACO algorithms have proven to be a powerful and easily adaptable tool which yields excellent results over highly combinatorial problems.

Most of the problems that have challenged ACO algorithms so far distinguish themselves by one characteristic feature – the objective function is easily calculated. Once a tour is chosen in the TSP or an assignment established for the QAP, the calculation of the quality of the solution requires very little time. The calculation of the objective function for the KAP, on the contrary, requires a significant amount of time.

Besides the modeling of the keyboard arrangement problem and the attempt to obtain optimized keyboard arrangements, the aim of this paper is to evaluate the suitability of ACO algorithms for such an optimization problem, involving a complex objective function. It may furthermore establish a new benchmark problem for comparing different types of metaheuristic, such as genetic algorithms, simulated annealing or taboo search.

It is worth to mention that other researches have been conducted concern-

ing the arrangement of letters on a keyboard but in a different direction. The problem then considers the situation of a keyboard where the number of letters exceeds the number of available positions and the objective is to minimize the inherent ambiguity (see [14, 21] for instances).

The paper is organized as follows. In section 2 an abstract model of a keyboard is introduced, which allows a representation of different types of keyboards, and the expression of heuristic ergonomic criteria is presented. It results in the definition of the KAP. After a bibliographic review of ACO algorithms, section 3 presents the solution algorithm. In section 4 computational results are given and the specificities of the algorithm as well as the quality of the obtained optimized keyboard arrangements commented on. The conclusion points out possible directions in which future research will be conducted.

2 The Keyboard Arrangement Problem

In order to define well this problem, the following approach is used. First, an abstract model of a keyboard is proposed enabling the representation of different types of keyboards. Second, an evaluation function for the ergonomic factors of a particular keyboard is defined, using heuristic criteria based on *Marsan's* work [19, 17, 18]. More details concerning these heuristic criteria can be found in [25] and in [15].

2.1 Representation of a keyboard

A keyboard basically serves to enter a string of characters into a computer by hitting a sequence of keys. An abstract definition therefore consists of a set of typeable characters and the respective sequences of keys which enable their construction. For instance, "E" is associated with the sequence of first hitting the "shift" key and then the key labeled "e". To this definition a geometrical representation has to be added in order to locate every key on the keyboard. Since most keyboards are arranged in rows and columns, we propose to represent the geographic position of a key by:

- a hand (left, right)
- a column (0-7 for the left hand and 0-8 for the right hand)
- 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 and French equivalents as well as the *Marsan* keyboard [19, 17, 18]. In addition, the following rule was used which is true for the standard typing systems on all presently available keyboards: A given column is always hit by the same finger. Since conventional keyboards dispose of a home row which

determines the location of the fingers in a relaxed position, this row serves as a reference row for calculations in optimization considerations.

Without any supplementary condition, there is an infinite number of possible layout solutions, any sequence of keys being a candidate for a letter. In order to simplify the problem by reducing the size of the solution space, the size and the structure of sequences are enforced. For example, the capital letter "E" is composed by hitting the shift key and the key which is used to produce the character "e". "E" and "e" therefore form a set which is subsequently called a combination character set (a table of the combination character sets may be found in [15]). It is the assignment of these combination character sets to keys which is addressed in the definition of the KAP. Besides, modifier keys like the "shift" or "AltGr" keys have a physically fixed position. These choices are justified by the ergonomic rule which demands a rapid mastery of the standard typing system and then by considerations of logical associations in terms of memorization. An assignment of a combination character set to a key is called an elementary assignment. A solution is therefore given by a complete set of elementary assignments, meaning that every combination character set is associated to a key.

An example for the transcription of the standard French AZERTY keyboard to the proposed model is given in figure 1.

2.2 Ergonomic criteria

A keyboard design should comply with four main objectives:

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

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 of 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. This justifies heuristic criteria introduced in the following paragraphs.

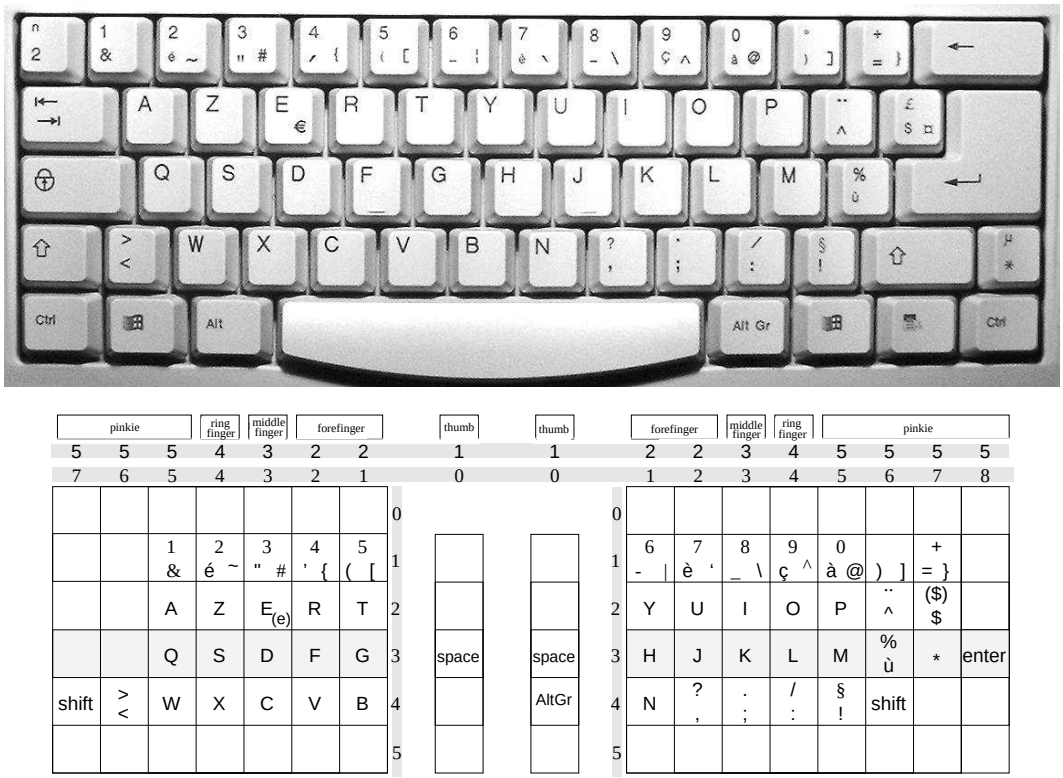

Figure 1: Representation of the *AZERTY* keyboard

2.3 The evaluation function

Because of the lack of exact ergonomic criteria, a set of heuristic rules established in [19], [18] and modified in [15] is used. They are supported by sets of rules established in [20] and [3]. The rules will not be explained in depth nor will a justification be given (see [15] for details).

Since the ergonomic criteria form two major groups, the notion of mono– and digraphs has to be introduced. A monograph is an isolated key which is struck in the process of typing a text. A digraph is a sequence of two consecutive keys hit in the process of typing a text. This idea allows the distinction between rules that concern the absolute position of a given key and the relative position of two consecutive keys.

Using a given keyboard layout, any text may be decomposed into a sequence of keys which have to be struck in order to type the text. This sequence may be seen as a sequence of monographs or a sequence of digraphs. It may therefore serve to establish a statistic of the appearance of mono– and digraphs. Subsequently, f_{m_i} and f_{d_i} stand for appearance frequencies of the monograph m_i and the digraph d_i, respectively, for a given keyboard layout and a given text. These frequencies serve as data for the evaluation of the

ergonomic criteria defined in the following.

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 has to be taken into account. In order to be able to evaluate a grade, an optimal distribution of the hit load has to be defined for each key position. 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. Each column then receives a ratio which takes into account the relative finger agility and endurance and each row receives a ratio representing the relative accessibility of that row (see table 1). The optimal load distribution $f_{m_i}^{opt}$ is then calculated by multiplying the three corresponding ratios for each key.

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

This ideal load distribution is given in figure 2 for the right hand. It

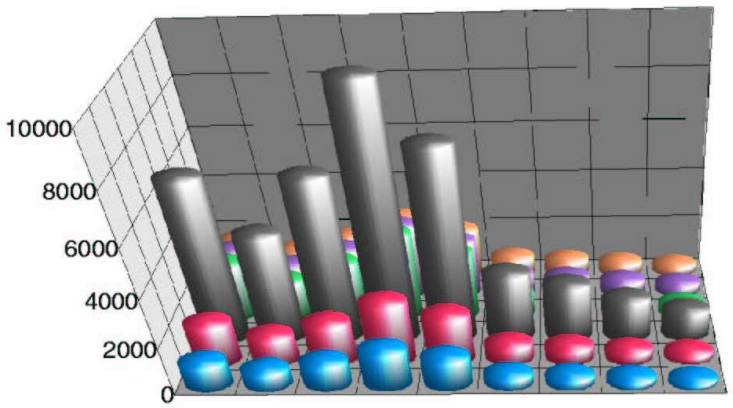

Figure 2: Ideal load distribution

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 height of the bars the ideal hit load normalized to have a maximum of 10000.

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.

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 enforcement of the structure of the sequences used to obtain the characters, which fixes the number of necessary keys. 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 consecutive 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 fingers of the same hand. 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 by fingers of the same hand.

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 and 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).

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

where $\kappa(d_i)$ is the weight coefficient. The chosen heuristic values for $\kappa(d_i) = \kappa(u, v)$ are represented in table 2, u and v 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 step 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, which allows a direct comparison with any other keyboard arrangement, the indicators v_j $(1 \leq j \leq 6)$ are used. 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. The values of γ_j are represented in table 3. 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.

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

Table 3: Weight coefficients γ_j

3 An ACO algorithm for the KAP

The KAP is a discrete, combinatorial 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 systems. Another important characteristic of the evaluation function should be stated: Compared to more classical problems as the traveling salesman problem or the quadratic assignment problem, the time necessary to calculate the global grade is excessive. An efficient algorithm should therefore not demand the 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, Ant Colony Optimization 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 these types of algorithms

to several different optimization problems like the TSP and the QAP finally led to the conclusion that their potential to solve the KAP might exceed the potential of other approaches.

Ant Colony algorithms were first introduced for solving combinatorial problems in [10]. Their fundamental idea 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 a simple agent that arranges the combination character sets on its own keyboard starting from a specific string of letters to distribute. In order to do so, it start with an empty keyboard, i.e. a keyboard whose keys are available to be assigned to any combination character set. Subsequently, the combination character sets which correspond to the letters in the string are assigned to the keys on the 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 2.3. Therefore the pheromone matrix acts as an abstract memory by indicating all pheromone levels that link a key to a combination character set. Once every ant has assigned all 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. The amount of pheromone that each ant places is proportional to the quality of its solution. The pheromone is then partially evaporated, the keyboards are cleared and the ants restart the search process with new character strings until a predefined stopping criterion is met.

3.1 Overview over the existing ACO algorithms

In the basic version of the Ant System algorithm [10], each ant proceeds by constructing elementary assignments by taking into account its apparent quality and the experience stored in the pheromone trails. Once each ant has found a complete solution, the quality of all solutions is evaluated and pheromone is distributed on the elementary assignments which are part of a complete solution, making sure that elementary assignments which are part of good solutions obtain more pheromone than those that are part of bad solutions.

This first version exhibited two major disadvantages: The excessive computational time and a clearly suboptimal convergence. In order to remedy these two weaknesses, two approaches have been proposed: An algorithm called

AS_{elite} in [10, 11] and one named AS_{rank} in [2].

AS_{elite} improved the convergence by emphasizing the best solution found. The modification of the basic version amounts to adding ants following the path of the best solution and modifying accordingly pheromone trails. By choosing as the best solution either the best solution found during one cycle of the algorithm or the best solution from the start of the algorithm, two different subversions have been developed, the best-of-cycle and the global best.

AS_{rank} introduces a new phase after each cycle of the algorithm. The solutions of all the ants having been evaluated, the ants are ranked according to the quality of their solutions. Only a fixed percentage p of the best ants is allowed to update the pheromone levels. This reduces the amount of necessary computational time in the phase when the pheromone is updated and accelerates the convergence of the algorithm.

A more radical approach to the modification of the algorithm was put forth in [23] by introducing $\mathcal{MAX}$–$\mathcal{MIN}$ Ant Systems. There, only the best ant is allowed to update the trails. In order to compensate for the thus emerging premature convergence, a minimum amount of pheromone τ_{min} is fixed. Once the pheromone deposited on a basic solution drops underneath this lower bound, it is reset to the minimum value. In order to make sure that a greater part of the space is searched, the algorithm initializes the pheromone levels to a preset maximum value τ_{max} which constitutes an upper bound for the trail intensity.

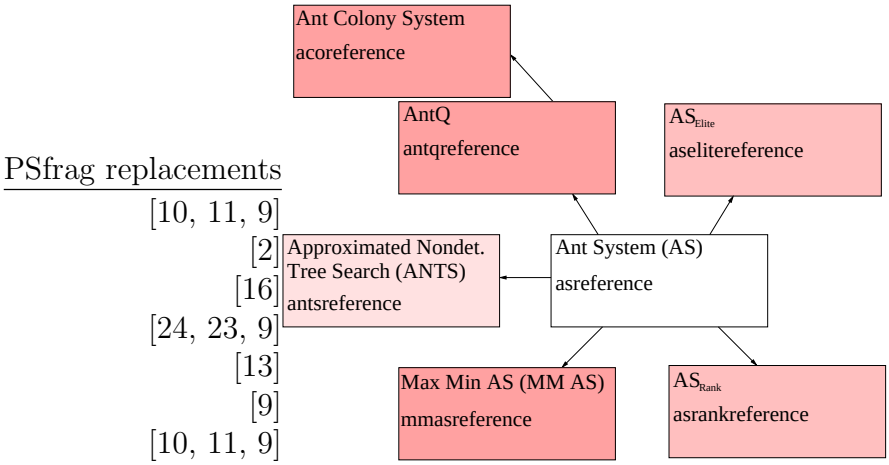

Figure 3: Overview over the different ACO algorithms

Another ant algorithm was derived from so–called Q-Learning which was introduced in [26]. The resulting algorithm, Ant–Q, was proposed in [13]. The new ideas consist of partially evaporating the pheromone of an elementary assignment immediately after its construction in order to speed up the search of the entire search space. It was the observation that two ants tend to build similar solutions once they start close to each other that led to this idea.

Evaporating the pheromone and thus creating a slight demotivation to follow the same path for the next ant leads to a broader search. A second feature is the modified rule for choosing the next elementary assignment. With a certain probability, the locally best possibility is chosen without taking the others into consideration. If this is not the case, the basic rule is used using the local and global quality of the solution. This results in a strategy which closely resembles the elitist approach. The third major change was the idea to use the prospective quality of the following elementary assignment to calculate the amount of pheromone deposited on a chosen partial solution. This concept was quickly abandoned for reasons of inefficiency. Concerning the choice of the ants which are allowed to deposit pheromone, Ant–Q follows the same strategy as $\mathcal{MAX}$–$\mathcal{MIN}$ Ant Systems: Only the best ant has this privilege.

The so far final and probably best performing algorithm was derived from Ant–Q by replacing the above mentioned concept by an implicitly guaranteed minimum of pheromone levels on each elementary assignment. The resulting algorithm was presented in [9] and named Ant Colony System.

In order to give a complete overview, the Approximate Non-deterministic Tree Search (ANTS) proposed in [16] should also be mentioned. Since the algorithm uses procedures which have already been developed for a standard problem like the TSP, but not yet known for the KAP, its application to the keyboard assignment problem does not seem very promising at present.

Figure 3 represents an overview of the above different algorithms.

3.2 Principles of the algorithm

The algorithm finally is a combination of several of the algorithms presented above. As a basic framework, we use the Ant Colony System algorithm since it displays the most promising feature. Besides, the basic ideas of AS_{Elite} are added by assigning more pheromone to the ant that found the best solution during one cycle. Thus, the importance of the best solution found is emphasized. $\mathcal{MAX}$–$\mathcal{MIN}$ AS also contributes to the final algorithm, since a minimum and a maximum values are defined for the pheromone level. Finally, AS_{Rank} principle is included by selecting the ants that are allowed to update the pheromone according to the respective ranking of their solutions.

Beyond the mere choice of certain characteristics of past algorithms, some other extensions are included. In the literature, one idea, which seems to have a promising impact on the characteristics of the algorithm, has been mentioned, but not yet thoroughly examined: A non-uniform initialization of the pheromone levels. In the standard version, the pheromone is initially distributed uniformly over all possible elementary assignments. However, 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, we have proposed to consider a non–uniform initialization of the pheromone matrix. This distribution – first suggested in [6] – translates an a–priori knowledge of the subspace in which the optimal solution may be found.

By attributing a higher pheromone level to the elementary assignments defined by a set of high quality keyboard arrangements than to the rest of the elementary assignments, the concentration of the search on the desired subspace may be implemented. The obvious inconvenience of this concentration 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 noteworthy modification is related 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, those sets are randomly assigned to the remaining keys.

3.3 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.

3.3.1 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.

3.3.2 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 produces a sequence of combination character sets that it places on its keyboard until the end of the string is reached. This procedure is described in detail in section 3.3.3.

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 4.

3.3.3 Character placement

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

1. The ant verifies if the combination character set has already been placed on the keyboard. If so, the ant skips the following steps and continues with the next combination character set.

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 2.3 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.

```
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 4: Solution algorithm for the KAP

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.

3.4 Indicator for the search activity

As stated before, ACO presents an example of an autocatalytic process. While the ants move in a large part of the search space in the beginning, they continually tend to concentrate on a subspace of smaller and smaller size. In order to describe the autocatalytic behavior and compare it to that observed when applying ACO to other combinatorial optimization problems, an indicator should be introduced. Such an indicator, which quantifies the effective size of the search space and therefore the search activity itself – the so–called

λ–branching factor – was defined in [10] and is recalled in the following definition:

Definition 1 *Let N_c be the set of all combination character sets and let N_t be the set of all keys. For $i \in N_c$, let $\tau_{i,max} = max\{\tau_{ij}, j \in N_t\}$ and $\tau_{i,min} = min\{\tau_{ij}, j \in N_t\}$ be respectively the greatest and the smallest level of pheromone of the possible elementary assignments of i. Furthermore, let $\delta\tau_i = \tau_{i,max} - \tau_{i,min}$ be their difference. For a scalar $\lambda \in [0, 1]$, the λ–branching factor $fact_\lambda(i)$ is defined as the number of elementary assignments associating the combination character set i to a key j, whose pheromone level satisfies the following equation:*

$$\tau_{ij} > \lambda \cdot \delta\tau_i + \tau_{i,min} \tag{11}$$

Hence, the λ–branching factor quantifies the number of elementary assignments for a fixed combination character set i which have a significant chance of being chosen by an ant. Since this search space description is limited to the combination character set i, another definition defining a global indicator is necessary.

Definition 2 *The mean λ–branching factor is the arithmetic mean of the λ–branching factors for all combination character sets i.*

$$\bar{fact}_\lambda = \frac{1}{N} \cdot \sum_{i \in N_c} fact_\lambda(i) \tag{12}$$

ACO being an autocatalytic process, the λ–branching factor naturally decreases with time, which basically meets the needs of the search activity. In the beginning, a relatively large space should be searched whereas in the later stages of the search, the effective search space should be concentrated around the optimal solution.

4 Computational Results

The experimental work consisted of three major phases. First, an efficient and effective parameter setting was determined. Then, the algorithm was run several times using this setting with and without initializing the pheromone matrix. Finally, the results and the developments of the λ–branching factor were examined.

4.1 Parameters setting

It is known that the convergence depends strongly on the parameters affecting the computational formula (see [10]). 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

[10] were chosen. Observing the development of the pheromone matrix, the parameter settings were adjusted experimentally. The best parameters found are summarized in table 4.

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 4: 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.

4.2 Initialization of the pheromone matrix

The algorithm was launched with and without an initialization of the pheromone matrix. A comparison showed that an initialization does significantly accelerate the convergence during the first few cycles, but does not enable to find better solutions or to reach the chosen solutions with notably more efficiency. Indeed, the pheromone matrix converges towards a certain limit which does not change much between runs and the total number of iterations needed to reach the chosen solution is very large. By way of illustration, figure 7 gives a good idea on the convergence of the solution.

4.3 Evolution of the λ–branching factor

As mentioned in section 3.4, the λ–branching factor is an adequate means to measure the development of the effective search space. The asymptotic behavior which is typical for autocatalytic processes can be observed. However, compared to the development of the λ–branching factor in a TSP application [13], it displays a much faster convergence and less regularity in the early stages (see figure 5).

4.4 Optimized keyboard arrangement

The best arrangement found for a French keyboard with the algorithm is presented in figure 6. It needed about 2000 iterations which corresponds to about 14 hours on a 400MHz PC. The development of the solution quality during the corresponding run is represented in figure 7. It leads to the satisfactory

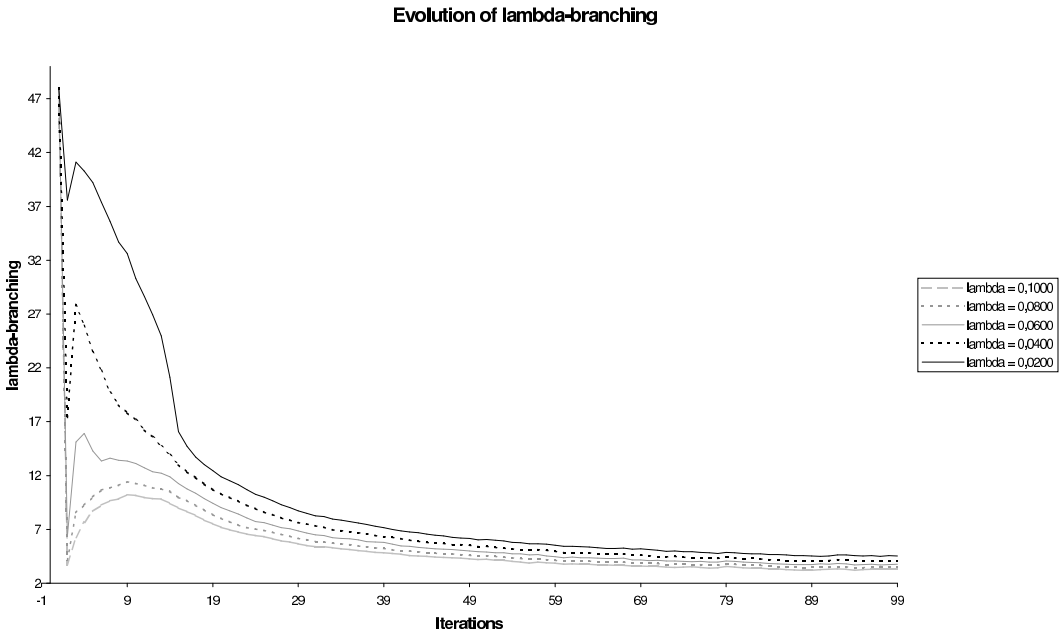

Figure 5: Evolution of the λ–branching factor – mean over ten test runs

global grade of 0.487, with the AZERTY keyboard as a reference. It also leads to a great improvement compared to the *Marsan* keyboard and outperforms it for all of the proposed evaluation criteria (see table 5).

Figure 6: The optimized arrangement for the French language

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, also found when starting from random pheromone matrices, is similar to those of *Dvorak* (in an English context) and *Marsan* manually optimized keyboards, which is intellectually satisfactory.

Experiments were also carried out to find arrangements of letters for a German and an English keyboards. The resulting layouts are presented in

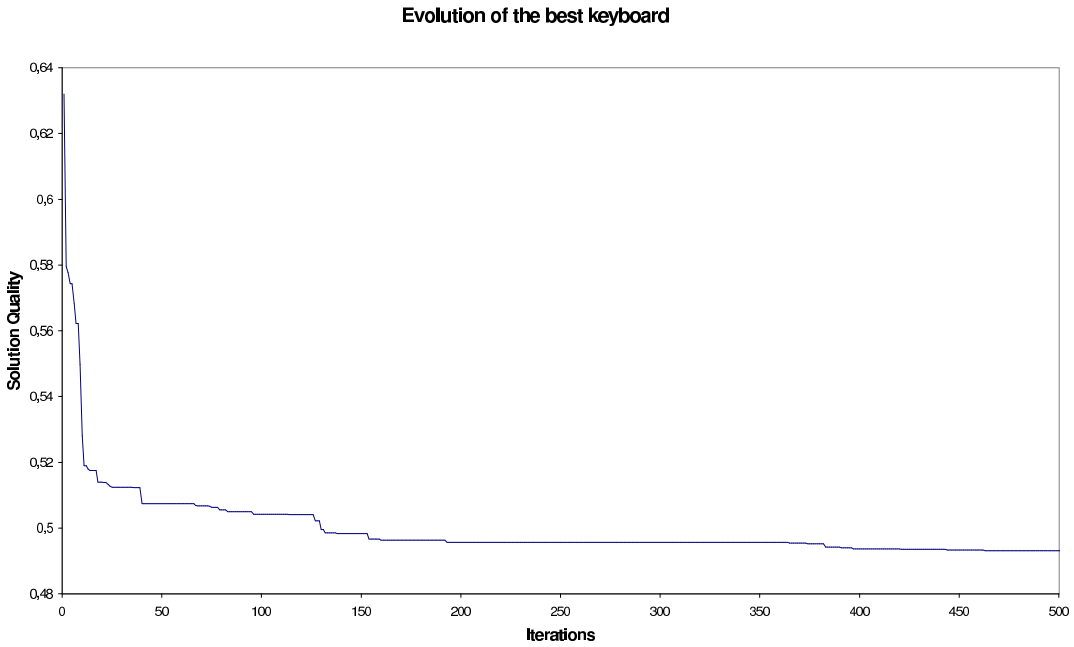

Figure 7: Evolution of the solution quality (French language)

	algorithm output	AZERTY	*Marsan*
load and accessibility	0.479	1	0.489
hand alternation	0.673	1	0.687
consecutive usage of the same finger	0.263	1	0.503
avoid big steps	0.222	1	0.315
hit direction	0.373	1	0.454
global grade	0.487	1	0.568

Table 5: The results in comparison to AZERTY and *Marsan* (French language)

figures 8 and 9. It also brings out the great possible improvements of standard keyboards: The global grade is 0.592 for the German keyboard, with the QWERTZ keyboard as a reference, while it is 0.593 for the English one, using the QWERTY keyboard as a reference. Compared to the *Dvorak* manually optimized English keyboard, whose global grade is 0.61, we only obtain a minor improvement, but let us mention that even if parameters are adjusted to the French language, the algorithm proves to slightly outperform *Dvorak* results.

Let us also mention that all these arrangements require six rows and therefore can not be mapped on the standard keyboard layout, but requires an ergonomic layout as presented in figure 10.

Figure 8: The optimized arrangement for the German language

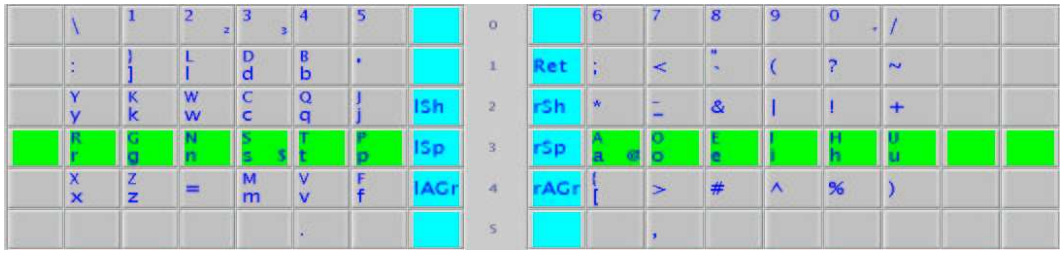

Figure 9: The optimized arrangement for the English language

5 Conclusions

In this paper, a new optimization problem that we call the Keyboard Arrangement Problem is addressed. It consists of finding the best possible arrangement of letters on a keyboard. A function is proposed for the evaluation of a keyboard with respect to a set of ergonomic rules. The paper does not focus on the design of the function, which has been approved by ergonomists and detailed in [15] and [25].

The presented algorithm for the solution of the KAP contains elements of several different ant optimization approaches found in literature. The obtained results show that the algorithm is effective in finding very good solutions. In

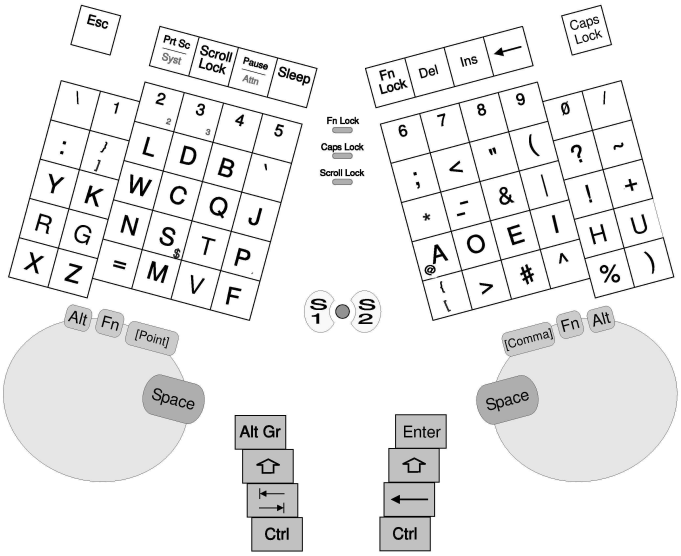

Figure 10: Ergonomic keyboard layout (with the English optimized solution)

particular, all known manually optimized keyboards such as the *Dvorak* and the *Marsan* keyboard were outranked.

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

Though no comparison to other metaheuristics has been carried out, the excellent results retrospectively justify the selection of ACO for the KAP. Future research will aim at such a comparison.

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] G. Di Caro and Marco Dorigo. AntNet: A mobile agents approach to adaptive routing. Technical Report 97-12, IRIDIA, Université Libre de Bruxelles, 1997.

[5] G. Di Caro and Marco Dorigo. AntNet: Distributed stimergetic control for communication networks. *Journal of Artificial Intelligence Research (JAIR)*, 9:317–365, 1998.

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

[7] D. Costa and A. Hertz. Ants can colour graphs. *Journal of the Operational Research Society*, 48:295–305, 1997.

[8] 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.

[9] Marco Dorigo, Gianni Di Caro, and Luca M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.

[10] 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.

[11] 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):29–41, 1996.

[12] L.M. Gambardella and M. Dorigo. HAS–SOP: A hybrid ant system for the sequential ordering problem. Technical Report 97-11, IDSIA, Lugano, CH, november 1997.

[13] Luca M. Gambardella and Marco Dorigo. Ant-Q: A reinforcement learning approach to the traveling salesman problem. In *Proceedings of ML-95, Twelfth Intern. Conf. on Machine Learning*, pages 252–260. Morgan Kaufmann, 1995.

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

[15] 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.

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

[17] 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 à écrire et similaires", Institut National de la Propriété Industrielle, France.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] Thomas Stützle and Holger H. Hoos. Ameisenalgorithmen zur Lösung kombinatorischer Optimierungsprobleme. Technical report, Technical University of Darmstadt, Department of Computer Science – Intellectics Group, Alexanderstr. 10, 64283 Darmstadt, Germany, 1998.

[25] M.O. Wagner, B. Yannou, S. Kehl, D. Feillet, and J. Eggers. Ergonomic modelling and optimization of the keyboard arrangement with an ant colony optimization algorithm. Technical Report CER 01-02 A, Laboratoire Productique Logistique, Ecole Centrale Paris, august 2001.

[26] Watkins. *Learning with delayed rewards*. PhD thesis, University of Cambridge, Psychology Department, University of Cambridge, England, 1989.