Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 148 (2003) 672–686
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

O.R. Applications

Optimization of the keyboard arrangement problem using an Ant Colony algorithm

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

Laboratoire Génie Industriel, Ecole Centrale Paris, Grande Voie des Vignes, 92295 Châtenay-Malabry Cedex, France

Received 11 October 2001; accepted 7 June 2002

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. It results in a new optimization problem that we name the keyboard arrangement problem. Based on the generic framework of Ant Colony optimization, an algorithm is developed and applied to this problem. New effective keyboard arrangements are deduced for several languages. Comparisons are made with standard manually optimized keyboards.

Keywords: 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 rearranged and thus adapted to other languages such as French and German. These counter-optimized keyboards were justified for some decades during which mechanical problems remained in typewriters. Rapidly it was not the case anymore but despite some serious work to get optimized keyboard arrangements (*Dvorak* in the USA and *Marsan* in France), the QWERTY layout remained the standard.

This fact might be partially explained by the drawbacks to a change in standard, which excludes repeated changes or changes with no guarantee of a prominent impact. As a matter of fact, ergonomic

* Corresponding author. Tel.: +33-1-41-131607; fax: +33-1-41-131272.
E-mail address: feillet@pl.ecp.fr (D. Feillet).

0377-2217/03/$ - see front matter
doi:10.1016/S0377-2217(02)00489-7

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 rules one can hope for therefore consist of a collection of heuristic rules which derive from common sense and are verified by experimental studies. Especially, this is the case of Marsan's ergonomic rules (1976) upon which are based our ergonomic rules.

Several proposals for a keyboard arrangement optimizing a criterion such as typing speed or rapid learning of the typing system were made (see Norman and Fisher, 1982). Furthermore, some papers (e.g., Pollatschek et al., 1975; Burkard and Offermann, 1977) investigated more complex ergonomic criteria, but did not insist on a realistic modeling nor an optimization method that would have finely addressed the problem. Thus, so far as we know, no paper has been devoted to the optimization of a realistic keyboard arrangement taking account of precise ergonomic criteria.

In this paper, we describe the application of a standard 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 (Wagner et al., 2001), designed from an ergonomic point of view, provides more extensive details on the definition of ergonomic criteria, their respective importance and the analysis of the results.

ACO algorithms have been used to solve very different types of optimization problems. Among these problems are the traveling salesman problem (TSP), see, e.g., Stützle and Dorigo (1999), the quadratic assignment problem (QAP), e.g., Dorigo et al. (1996), the sequential ordering problem (Gambardella and Dorigo, 1997), the graph coloring problem (Costa and Hertz, 1997) and several other combinatorial, static optimization problems as well as dynamic ones which are primarily found in the routing for telecommunication networks (e.g., Di Caro and Dorigo, 1997, 1998). ACO algorithms have proven to be a powerful and easily adaptable tool which yields excellent results for 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 KAP 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 furthermore establishes a new benchmark problem for comparing different types of metaheuristic, such as genetic algorithms, simulated annealing or taboo search.

It is worth mentioning that other research is conducted concerning the arrangement of letters on a keyboard but in a different direction. Other researchers consider the case of a keyboard where the number of letters exceeds the number of available positions and the objective is to minimize the inherent ambiguity (see Glover, 1986 or Oommen et al., 1992 for instance).

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 brief 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 correctly define 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. To this

aim we use a set of heuristic criteria issued from Marsan's work (1976, 1979, 1987) and that have been validated through heavy experiments carried out on both conventional and Marsan optimized keyboards. More details concerning these heuristic criteria can be found in Wagner et al. (2001) and Kehl and Wagner (2000).

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 can therefore be the following: 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. 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 Kehl and Wagner (2000)). It is the assignment of these combination character sets to keys which is addressed in the definition of the KAP. Besides, modifier keys as the shift 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.

A supplementary important feature of the model is related to modifier keys as shift and to the "space" key. It is conventional to provide two possibilities, that is two keys, for the hit of a shift. In the same way, the usual space bar can be hit by any of the two thumbs and is represented in our model by two different keys, each one being associated with one thumb. It introduces new difficulties since a typist has to decide which one of the two keys he is going to use. We assume that some deterministic decision rules are applied. A shift key is struck with the opposite hand, compared to the hand that strikes the following key. The space key is struck by the opposite hand, compared to the hand that strikes the previous key.

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

2.2. Ergonomic criteria

A keyboard design should seek to attain 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.

Fig. 1. Representation of the AZERTY 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 basic difficulty can be encountered in the different types of fatigue and their potential pathological nature. Short-term fatigue reduces typing speed but does not present a danger to the writer's health whereas long-term fatigue might lead to repetitive stress disorders. A second uncertainty lies in the localization of fatigue. 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 fatigue. This justifies the need of heuristic criteria, introduced in the following section.

2.3. The evaluation function

In order to evaluate the quality of the arrangement of a keyboard, we use a set of six heuristic criteria. These criteria were first established by Marsan (1976, 1987), but modified in Kehl and Wagner (2000). They are supported by sets of rules established in Norman and Fisher (1982) and Burkard and Offermann (1977). The rules will not be explained in depth nor will a justification be given (see Kehl and Wagner, 2000 for details). For a given keyboard arrangement, a score is associated with each criterion and a global score is computed by a simple weighted linear combination of the six scores.

Since the rules 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 heuristic criteria which are defined now. For each of these criteria, we shortly describe the ergonomic rules that support it and we formulate the equation that provides the score of a keyboard arrangement regarding this criteria.

2.3.1. Accessibility and load

The combination character sets should be distributed on the keys in a way that the total load is shared adequately by all fingers. The fact that some keys are less accessible than others has to be taken into account. So as to evaluate a score, 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 in their performance and endurance. Since the two ergonomists we consulted did not agree on giving the right hand an advantage over the left hand, the ratio is chosen to be 50% for each hand. Each column then receives a ratio which takes into account the relative agility and endurance of the fingers and each row receives a ratio representing the relative accessibility of that row (see Table 1). The optimal load distribution f_{mi}^{opt} is then calculated by multiplying the three corresponding ratios for each key.

Table 1
Ideal load distribution

No.	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

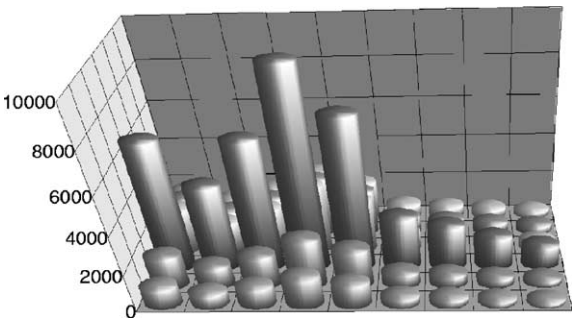

Fig. 2. Ideal load distribution for the right hand.

This ideal load distribution is given in Fig. 2 for the right hand. 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 height of the bars the ideal hit load normalized to have a maximum of 10,000.

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

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

where Ξ_1^m is the set of all monographs.

2.3.2. 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 score v_2 is therefore calculated by dividing the number of characters in a text by the number of keystrokes necessary to produce the text. While it is relatively important for the most general definition, this score 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 Alden et al., 1972).

2.3.3. 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 rule, we sum the frequency of the digraphs which are typed by fingers of the same hand. In terms of mathematical formulation the score 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.

2.3.4. Consecutive usage of the same finger

The preceding alternation rule is also true for the fingers. Two consecutive keys should not be hit by the same finger. The criterion 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 inconvenience for 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:

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

The chosen distance function is the so-called Manhattan distance given by:

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

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.

2.3.5. Avoid big steps

When one hand is used for two consecutive hits, great distances which require awkward hand postures 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).

Table 2
Big step coefficients

	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

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

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.

2.3.6. Hit direction

For digraphs typed by one hand only, the preferred hit direction is from the little finger towards the thumb. This is the natural finger movement for most 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 score is given by:

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

2.3.7. Global score

To define a global score, which allows a direct comparison with any other keyboard arrangement, the scores v_j $(1 \leqslant j \leqslant 6)$ are combined linearly. Actually, since these scores have different ranges and units and since they have different relative importances, they are first divided by the respective scores of a reference keyboard $v_{j,\mathrm{ref}}$. Once the scores are turned dimensionless they are multiplied by a relative weight coefficient γ_j and summed. The values of γ_j are represented in Table 3. These values were obtained with the help of two

Table 3
Weight coefficients γ_j

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

specialized ergonomists, using a pairwise comparison method (see Limayem and Yannou, 2002). This leads to the final formula:

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

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

3. An ACO algorithm for the KAP

Ant Colony algorithms are now classical approaches for solving combinatorial optimization problems, that were first introduced by Dorigo et al. (1991). Their fundamental idea is based on the behavior of natural ants who succeed in finding the 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 a 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 the proximity of the nest and of the food source respectively into account. 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 starts 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 successful placements in the past into account and respecting the ergonomic rules established in Section 2.3 as much as possible. 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 leave pheromone on the elementary assignments which are part of its solution. The amount of pheromone that each ant deposits 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.

The a priori interest of a such approach for the KAP, compared to other metaheuristic approaches, relies on the complexity of the evaluation of a key board arrangement. The important point is that this computation first requires the computation of the mono- and digraph statistic, which depends of the solution. With the ACO approach, when an ant arranges letters on a keyboard, it simultaneously computes the statistic, which permits a fast evaluation of the solution at the end of the cycle. This particularity and the successful application of ACO algorithms to several different optimization problems, especially the QAP that offers many similarities with the KAP, finally led to the conclusion that their potential to solve the KAP might exceed the potential of other approaches.

3.1. Principles of the algorithm

Since the appearance of the first ACO algorithm, many approaches have been proposed to enhance its efficiency. Our algorithm presents a combination of some of these approaches. As a basic framework, we use the Ant Colony System

algorithm described in Dorigo et al. (1999). The basic ideas of AS_{Elite} (Dorigo et al., 1991, 1996) 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 (Stützle and Hoos, 1996) also contributes to the final algorithm, since a minimum and a maximum values are defined for the pheromone level. Finally, AS_{Rank} principle (Bullnheimer et al., 1997) is included by selecting the ants that are allowed to update the pheromone according to the respective ranking of their solutions. The reader is referred to the indicated papers for more details about these basic features.

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 Corne et al. (1999)—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 drawback 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 impossible 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.

3.2. Description of the algorithm

From the principles mentioned above, the following algorithm has been deduced. It 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 predefined number of cycles.

3.2.1. 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 an a priori knowledge of the solution in the form of high quality keyboard arrangements into account.
6. The algorithm parameters (see definitions later) are initialized.

3.2.2. Iteration

The second part of the algorithm is repeated until a certain number of iterations have 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 3.2.3.

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}:

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

7. A number p of the best solutions are selected and pheromone is updated according to their rank: The kth best ant, with $1 \leqslant k \leqslant p$, 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 Fig. 3.

3.2.3. 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.

```
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;
```

Fig. 3. Solution algorithm for the KAP.

2. For all free keys j, the ant calculates a probability according to the random-proportional decision rule given in Eq. (9):

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

where Ω is the set of not yet occupied keys and η_{ij} the global score 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 score η_{ij} only requires 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.

3.3. 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 can be introduced. Such an indicator, the so-called λ-branching factor, was defined in Dorigo et al. (1991). It quantifies the effective size of the search space and therefore the search activity itself. It 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 $\mathrm{fact}_\lambda(i)$ 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:

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

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.

$$\overline{\mathrm{fact}}_\lambda = \frac{1}{N}\sum_{i\in N_c}\mathrm{fact}_\lambda(\mathrm{i}) \tag{12}$$

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.

The evolution of the λ-branching factor is closely related to the value of all the parameters that are defined within the algorithm. Actually, it would be a valuable indicator for the adjustment of these parameters, possibly in a dynamic fashion. For the time being, however, we simply use it to determine the size of the search space throughout the resolution.

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 Dorigo et al., 1991). 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 Dorigo et al. (1991) were chosen. Observing the development of the pheromone matrix, the parameter settings were adjusted experimentally. The best parameters found are summarized in Table 4.

Since the objective is to find 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. We determined values of the parameters that appear to be efficient, in the sense that slight changes in some parameters frequently led to a significant decrease of the solution quality. However, it has to be said that a detailed statistical examination of different parameter settings was left out.

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, Fig. 6 gives a good idea on the convergence of the solution.

Table 4
Choice of optimization parameters

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

4.3. Evolution of the λ-branching factor

As mentioned in Section 3.3, the λ-branching factor is an adequate means to measure the development of the effective search space all through the solution. Fig. 4 describes its evolution, for different values of λ. 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, as observed in Gambardella and Dorigo (1995), it displays a much faster convergence and less regularity in the early stages.

4.4. Optimized keyboard arrangement

The best arrangement found for a French keyboard with the algorithm is presented in Fig. 5. It required about 2000 iterations which corresponds to about 14 hours on a 400 MHz PC. The source for the text which is parsed during the algorithm is taken from the newspaper Le Monde. The development of the solution quality during the corresponding run is represented in Fig. 6. It leads to the satisfactory global score 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).

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.

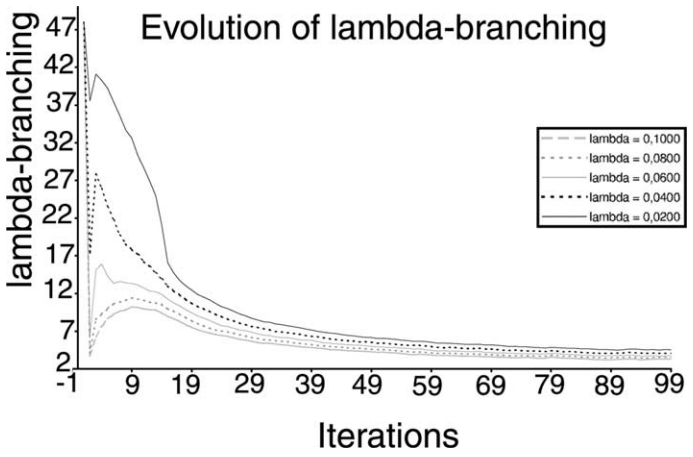

Fig. 4. Evolution of the λ-branching factor—mean over 10 test runs.

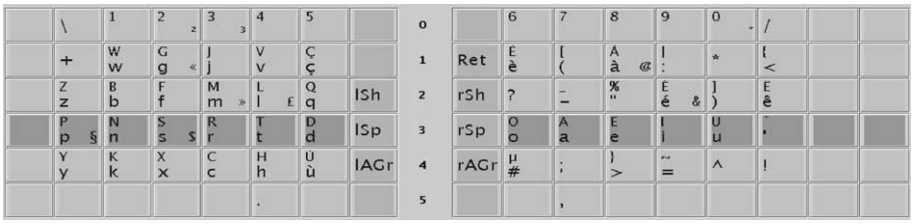

Fig. 5. The optimized arrangement for the French language.

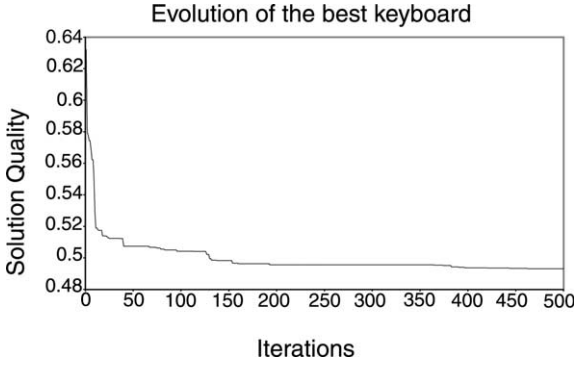

Fig. 6. Evolution of the solution quality (French language).

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.

Table 5
Algorithm output and Marsan scores in comparison to AZERTY (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 score	0.487	1	0.568

Experiments were also carried out to find arrangements of letters for a German and an English keyboards. Text sources are then respectively Der Spiegel and USA Today. The resulting layouts are presented in Figs. 7 and 8. It also brings out the possibility of major improvements to standard keyboards: The global score is 0.592 for the

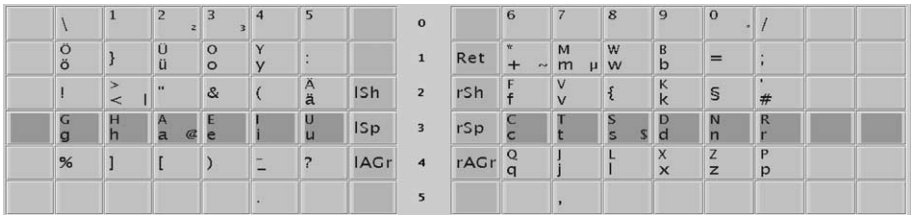

Fig. 7. The optimized arrangement for the German language.

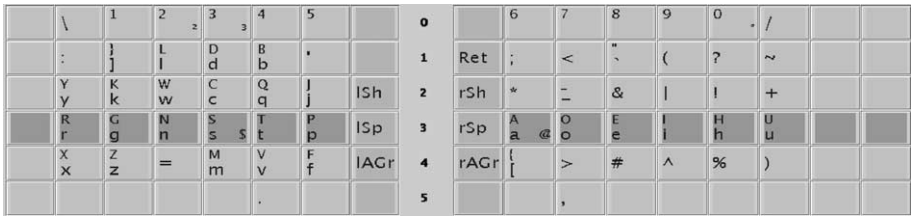

Fig. 8. The optimized arrangement for the English language.

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 score is 0.61, we only obtain a minor improvement. Note however that these results are obtained with parameters adjusted to the French language.

Let us also mention that all these arrangements require six rows and therefore cannot be mapped on the standard keyboard layout, but requires an ergonomic layout as presented in Fig. 9.

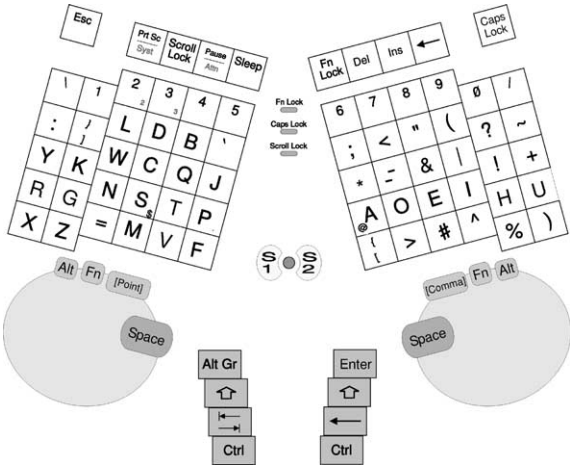

Fig. 9. Ergonomic keyboard layout (with the English optimized solution).

5. Conclusions

In this paper, a new optimization problem that we call the KAP is addressed. It consists of finding the best possible arrangement of letters on a keyboard. A generic modeling and a function are proposed for the evaluation of a keyboard with respect to a set of ergonomic rules.

The presented algorithm for the solution of the KAP contains elements of several different ant optimization approaches found in the literature. The computational experiments show that the algorithm is effective in finding very good solutions. In particular, all known manually optimized keyboards such as the Dvorak and the Marsan keyboard were outranked. By the time being, no extensive field trials have been performed to evaluate the benefits that would yield these new keyboards, even if some specialists of keyboard arrangements and users of Marsan keyboards recognize the quality of our results. The main reason is that it is still very difficult to evaluate the true advantages that would provide a new keyboard.

Actually, the keyboards that we propose do not aim to become new reference keyboards in the

future. The reality of typewriting is far from being completely known and extensive studies have to be performed by ergonomists to attain a deep understanding of the rules that should drive the arrangement of letters on a keyboard. However, this work provides them a valuable simulation tool and proves the efficiency of modern optimization algorithms for this problem. Furthermore, it permits or would permit to easily compute efficient keyboards in different contexts, if needed. Examples are keyboards for right-handed or left-handed people, keyboards for multilingual usage or keyboards adapted to a specific purpose (with a repetitive use of some specific symbols).

Furthermore, this paper demonstrates the suitability of ant algorithms for problems such as the KAP, for which the evaluation of a solution requires a complex set of operations. The special feature of the algorithm is that the evaluation and the construction of a solution by an ant can be performed simultaneously. Hence, it gives an important advantage compared to metaheuristics based on local search, for which the evaluation of too many solutions would be detrimental.

Finally, it is interesting to see that, although we have restricted ourselves to computer keyboards, our approach could easily be adapted to other kinds of keyboards, as the ones that are used for industrial control.

Acknowledgements

This paper has benefitted from helpful comments and suggestions by three anonymous referees, who are gratefully acknowledged.

References

Alden, D.G., Daniels, R.W., Kanarick, A.F., 1972. Keyboard design and operation: A review of the major issues. Human Factors 14 (4), 275–293.

Bullnheimer, B., Hartl, R.F., Strauß, C., 1997. A new rank based version of the ant system–a computational study. Technical Report POM-10/97, Institute of Management Science, University of Vienna.

Burkard, R., Offermann, J., 1977. Entwurf von Schreibmaschinentastaturen mittels quadratischer Zuordnungsprobleme. Zeitschrift für Operations Research 21, B121–B132.

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

Di Caro, G., Dorigo, M., 1998. AntNet: Distributed stimergetic control for communication networks. Journal of Artificial Intelligence Research (JAIR) 9, 317–365.

Corne, D., Dorigo, M., Glover, F., 1999. New Ideas in Optimization. McGraw-Hill, NY.

Costa, D., Hertz, A., 1997. Ants can colour graphs. Journal of the Operational Research Society 48, 295–305.

Dorigo, M., Caro, G.D., Gambardella, L.M., 1999. Ant algorithms for discrete optimization. Artificial Life 5 (2), 137–172.

Dorigo, M., Maniezzo, V., Colorni, A., 1991. Ant system: An autocatalytic optimizing process. Technical Report 91-016, Dipartimento di Elettronica e Informazione, Politechnico di Milano, Italy.

Dorigo, M., Maniezzo, V., Colorni, A., 1996. The ant system: Optimization by a colony of cooperating agents. IEEE Transaction on Systems, Man and Cybernetics—Part B 26 (1), 29–41.

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

Gambardella, L.M., Dorigo, M., 1995. Ant-Q: A reinforcement learning approach to the traveling salesman problem. In: Proceedings of ML-95, Twelfth Intern. Conf. Machine Learning. Morgan Kaufmann, pp. 252–260.

Glover, D.E., 1986. Experimentation with an adaptive search strategy for solving a keyboard design/configuration problem. PhD thesis, University of Iowa.

Kehl, S., Wagner, M.O., 2000. Modélisation et optimisation de la répartition des lettres sur un clavier d'ordinateur. Applied scientific project, Laboratoire Productique Logistique, Ecole Centrale Paris.

Limayem, F., Yannou, B., 2002. Le tri croisé de Monte Carlo: une boîte à outils pour l'aide à la décision coopérative. To appear in Revue de CFAO et Informatique Graphique.

Marsan, C., 1976. 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.

Marsan, C., 1979. 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.

Marsan, C., 1987. 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.

Norman, D.A., Fisher, D., 1982. Why alphabetic keyboards are not easy to use: Keyboard layout doesn't much matter. Human Factors 24 (5), 509–519.

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

Pollatschek, M., Gershoni, M., Tadday, Y., 1975. Improving the Hebrew typewriter. Technical report, Haifa.

Stützle, T., Dorigo, M., 1999. ACO algorithms for the traveling salesman problem. Technical report, IRIDIA, Université de Bruxelles, Belgium.

Stützle, T., Hoos, H.H., 1996. Improving the ant system: A detailed report of the MAX-MIN ant system. Technical Report AIDA-96-12, Technical University of Darmstadt, Department of Computer Science-Intellectics Group, Germany.

Wagner, M., Yannou, B., Kehl, S., Feillet, D., Eggers, J., 2001. 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. Available at <http://www.pl.ecp.fr/~feillet>.