International Journal of Industrial Ergonomics 41 (2011) 43–52

Contents lists available at ScienceDirect

International Journal of Industrial Ergonomics

journal homepage: www.elsevier.com/locate/ergon

Cyber Swarm optimization for general keyboard arrangement problem

Peng-Yeng Yin*, En-Ping Su

Department of Information Management, National Chi Nan University, 470 University Road, Puli, Nantou 54561, Taiwan

ARTICLE INFO

Article history:
Received 23 August 2009
Received in revised form
13 October 2010
Accepted 4 November 2010
Available online 4 December 2010

Keywords:
Cyber swarm optimization
Keyboard arrangement problem
Multiple objectives
Metaheuristics
Optimization

ABSTRACT

Traditional QWERTY keyboard has been known to be an ill design due to its low typing speed, fatigue-inducement, no reference to learnability, to name a few. Many researchers have put their effort on optimizing the character arrangement of the keyboard by taking into account the co-occurrence frequency of characters in words, typing ergonomics, and word disambiguation effectiveness. However, most of the existing works addressed the problem for a single objective instead of multiple ones, and the design of customized keyboards for motor-impaired users is often overlooked. In this paper, we propose a general mathematical model that integrates varying keyboard arrangement context. A Cyber Swarm method considering multiple objectives is proposed and it can create additional benefits that cannot be obtained by traditional methods. Our Cyber Swarm method accommodates ergonomic criteria and disambiguation/prediction effectiveness simultaneously. Experimental results manifest that the Cyber Swarm keyboard outperforms several benchmark keyboards and other competing algorithms. We also show an illustrative example for preliminary keyboard shape design which could be very useful in customized keyboard production for motor-impaired users whose physical capacity has been evaluated a priori. Finally, an empirical experiment involving human subjects is performed in order to analyze the feedbacks from humans' experiences in using the new keyboard layouts.

Relevance to industry: Using the proposed general framework for keyboard arrangement, various scenarios (for example, single-character or multi-character keys, single-finger or multi-finger typing) raised in industrial production can be all taken into account without the hassle to change solution methods for different settings. Hence, the time and costs in industrial production for customized keyboard is significantly reduced.

1. Introduction

The advance of technologies in speech and handwritten character recognition does not replace the use of text entry keyboards; instead, the demand of various keyboards for particular objectives and working scenarios is even increasing. Despite of the standard computer keyboards which accommodate single character in a key, the keypads embodied in mobile phones and personal digital assistants (PDAs) usually have to arrange multiple characters in a key due to smaller typing areas. In some circumstances, virtual keyboards displayed on touch screen are more preferable such as implementations in kiosks or pilot's cockpit. For the motor-impaired users, customized text entry devices are needed to take physical disabilities into account.

The objectives of the designed keyboards used in different working scenarios could be very distinct. They can be classified into subclasses using two dimensions. The first dimension characterizes keyboards with *single-character* or *multi-character* key, while the second dimension models the user keying behavior as *single-finger* or *multi-finger* typing. The single-character keyboard (SCK) exactly locates one character on each key, this form of keyboard arrangement has predominated the market for a long time. The prevailing SCK is the QWERTY keyboard which was developed in 1873, however, it was not designed to increase the keying speed but to slow it down in order to lengthen the life duration of the text entry device. Conspicuously, ergonomic criteria were not considered in the design of QWERTY keyboard and it could cause fatigue easily and may result in musculature injuries in the long run. The first successful ergonomically designed text entry device may be the Dvorak keyboard (Cassingham, 1986) developed in 1932 which considered several human–computer factors such as home row usage, co-occurrence frequency of consecutive letters in English words, and easiness of typing learning. Light and Anderson (1993) formulated the keyboard arrangement as a combinatorial optimization problem and applied the simulated annealing technique to minimize the total finger

* Corresponding address. Tel.: +886 49 2910960; fax: +886 49 2915205.
E-mail address: pyyin@ncnu.edu.tw (P.-Y. Yin).

0169-8141/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ergon.2010.11.007

traveling time during the typing. Eggers et al. (2003) used a more sophisticated model and precisely defined six ergonomic criteria, namely, the tapping load distribution, number of keystrokes, hand alternation, finger alternation, finger posture, and hit direction. Deshwal and Deb (2003) used Eggers' criteria as the performance indices and developed a near-optimal Hindi keyboard.

The multi-character keyboard (MCK) arranges more than one characters on a single key, this design is motivated by the need to facilitate language production using a telephone keypad whose working area is smaller than that of a standard keyboard and thus affords a fewer number of keys. To disambiguate each key, the first proposal simply groups the letters alphabetically and requires the typist to produce a single letter by two keystrokes. The fist keystroke selects the group the target letter belongs to, and the second chooses the letter index in that group. Later, as mobile devices with LCD screen appear, more intelligent disambiguation methods are proposed. Word-level disambiguation technique (Minneman, 1985; Oommen et al., 1991; Sorensen, 2007) identifies which combinations of the input letter sequence actually correspond to meaningful words in dictionary and allows the typist to select from the candidates, so the main purpose is to find the optimal keyboard arrangement such that the number of dictionary words that has the same keystroke sequence with others is minimized. However, it is inconvenient when typing user-defined words such as slang, jargon and acronyms. By contrast, character-level disambiguation algorithm (Levine et al., 1987; Foulds et al., 1987; Oommen et al., 1991; Arnott and Javed, 1992; Lesher et al., 1998; Tanaka-Ishii et al., 2002) immediately predicts the most probable character after each keystroke based on *ngrams* statistics, which is derived from the co-occurrence frequency of n consecutive characters in the language. If the predicted letter is not the intended one, the typist strikes a retry key to select the next most probable character. The objective of character-level disambiguation is to reduce the total number of necessary keystrokes.

The scenario of using a single-finger keyboard (SFK) arises when performing typing with hand-held devices such as mobile phones or PDAs, using virtual keyboards displayed on touch screen, or when the user is motor-impaired and needs to use a point-based text entry means (e.g., head, eye, or stylus). The design of an SFK involves three major concerns, namely, movement time between two target keys, character transition frequency, and keyboard shape. There have been several alternative SFKs developed on this purpose. These include alphabetical layout, FITALY, OPTI, Metropolis, Hooke, and ATOMIK (Mackenzie and Soukoreff, 2002; Zhai et al., 2002a,b). Due to the increasing popularity of mobile phones with a touch screen, the key size and the key location affecting the activation area and input accuracy are of major concern (Park and Han, 2010a,b; Kwon et al., 2009; Lee, 2010). By contrast, the design of a multi-finger keyboard (MFK) is focused to reduce the risk of causing musculature injuries and to enhance the posture comforts of long-term typing (Eggers et al., 2003; Tomatis et al., 2009). Attention is also drawn to increase the typing speed and improve the learnability of the keyboard arrangement.

The original contribution of this paper includes the following aspects. (1) We propose a general keyboard arrangement model that accommodates varying objectives required by different working scenarios (SCK, MCK, SFK, and MFK) in a single framework. Traditional keyboard arrangement models are customized to a particular working scenario and may overlook the advantages of useful keyboard arrangement methods adopted in other scenarios. On contrary, we alleviate the hassle for designing another mathematical model each time a new application arises, the user only needs to specify the weights for various designing objectives and our algorithm will automatically generate the optimized keyboard arrangement fitting to the desired scenario. (2) We propose to use a metaheuristic approach (the Cyber Swarm Algorithm) which is an optimization meta-strategy and is not customized to any particular application, so our method is adaptable to different applications. Most of the existing keyboard arrangement approaches are application-specific heuristics which are hardly to be adapted to another application. For example, the Dvorak keyboard relying on heavy home row usage is clearly not suitable for the disabled who have difficulties in using certain fingers. Also, the 3×3 Alphabetic keyboard arranges the 26 English letters alphabetically on the nine keys in a raster-scan order for easy learnability. However, the alphabetic arrangement heuristic is not appropriate for applications with a greater number of keys because the majority of keystrokes may be done in just a small region of the layout, reducing the typing speed and causing severe finger strain. (3) Our unified model combining the objectives for different scenarios can create extra valuable benefits. For example, the prediction algorithm embedded in MCK for disambiguation can be applied to SCK for predicting the intended word based on the entered prefix, and thus reduce the number of keystrokes (Note that all current SCK keyboards assume that the users have to type all letters contained in the intended word). In addition, the Fitts' (1992) Law for estimating the typing cost in the SFK design based on inter-key distance, and the ergonomic criteria (such as alternation, posture, and sequence of finger during the typing) considered in the MFK design can be unified in a general framework by a penalty matrix which is also applicable to the keyboard arrangement customized to the disabled. (4) Our keyboard arrangement method can be used as a preliminary step of customized keyboard shape design for motor-impaired users when the information of awkward movement is available. (5) In addition to theoretical analysis, we conducted an empirical experiment involving human subjects. From the human–computer interaction (HCI) perspective, our Cyber Swarm keyboard layout can be learned quickly and effectively by traditional QWERTY users, the typing rate almost doubles after a reasonable amount of practice.

The keyboard arrangement problem (KAP) can be polynomial-reduced to the quadratic assignment problem (Ladany, 1975) which is NP-complete. Some of the existing solutions to the KAP used heuristics such as n-opt (Lesher et al., 1998), entropy minimization (Tanaka-Ishii et al., 2002), and multi-start descent algorithm (Sorensen, 2007). Alternating proposals applied metaheuristic techniques including genetic algorithms (Deshwal and Deb, 2003; Minneman, 1985; Oommen et al., 1989), simulated annealing (Light and Anderson, 1993; Li et al., 2006), and ant colony optimization (Eggers et al., 2003; Wagner et al., 2003). In this paper, we adopt the Cyber Swarm metaheuristic which is inspired from the particle swarm optimization to solve the keyboard arrangement problem. To expedite the computation for particle fitness evaluation, a useful bounding technique is proposed.

The remainder of this paper is organized as follows. Section 2 defines the addressed problem. Section 3 presents the proposed method for the KAP problem. Section 4 reports the comparative performances and analysis. Finally, Section 5 concludes this work.

2. General keyboard arrangement problem

The purpose of a general keyboard arrangement problem (GKAP) which allocates M characters to N keys is to find a keyboard arrangement that optimizes a set of specified objectives with a disambiguation/prediction algorithm. There is no definitive relationship between the values of M and N, however, it could be analogous to the SCK scenario if $M = N$, and to the MCK design if $M > N$. It is also useful to allow $M < N$ resulting in a keyboard arrangement using only a subset of available keys, this could serve as a preliminary step for customized keyboard shape design for motor-impaired users when the information regarding physical disabilities is known a priori.

The objectives of a GKAP can be classified into two categories. The first category addresses the functionality according to

ergonomic criteria while the second group aims at maximizing the effectiveness of disambiguation/prediction. We consider both categories of objectives and they are described as follows.

2.1. Ergonomic criteria

Eggers et al. (2003) used six Marsan's ergonomic criteria (tapping load distribution, number of keystrokes, hand alternation, finger alternation, finger posture, and hit direction) in designing an SCK which was subsequently adopted in designing a Hindi keyboard by Deshwal and Deb (2003). Based on the six Marsan's ergonomic criteria, we present two ergonomic performance indexes that is applicable in general keyboard arrangement context.

2.1.1. Key accessibility

Eggers et al. (2003) argued that typists have different accessibility to separate keys depending on the locations. The optimal keyboard arrangement should produce a tapping load distribution that is close to an ideal one when typing a text. Eggers et al. (2003) proposed the ideal load distribution for general users by consulting two ergonomists. Eggers' ideal load distribution also considers the middle row as the home row and assumes faster typing speed and less strain to the fingers would be resulted by respecting the ideal load distribution where 43.5% of the key strokes are done in the home row. We propose to broaden this ergonomic factor by including the accessibility of motor-impaired users. In general, the typing means may involve single-finger (or using a wearable stylus) or multi-finger. An ideal load distribution should be determined by consulting ergonomists as well as physicians who are acquainted with the particular user's physical conditions. Imaging how significant the difference of accessibility is between a user moving his head to control a stylus and a user typing with multiple fingers. So the ideal load distribution is user-dependent. In our formulation, the objective with respect to the key accessibility for keyboard arrangement k is defined as follows.

$$\text{Min } O_1(k) = \left\| l_{kp}(\pi) - l^* \right\|, \tag{1}$$

where $l_{kp}(\pi)$ is the load distribution produced by typing a well-represented text π using keyboard arrangement k with the prediction algorithm p, and l^* is the ideal load distribution which is represented as an N-dimensional vector whose i-th component indicates the ideal tapping load on the i-th key. That is, the best key arrangement embedding a given prediction algorithm is the one that minimizes the norm distance between the produced load distribution $l_{kp}(\pi)$ and the ideal load distribution l^* for the particular user. It is noted that the key accessibility focuses on the overall tapping load distribution on separate keys, no attention is given to the sequence of the keys used to do the typing.

2.1.2. Posture comfort

The typing sequence of fingers inducing awkward postures would cause musculature injuries and impair the typing speed. Eggers et al. (2003) analyzed the impact of hand alternation, finger alternation, finger posture, and hit direction, and gave a respective penalty function for each of the factors to avoid unpreferred usage of fingers. It is not difficult to transform Eggers' penalty functions into one unified penalty coefficient matrix as given in Table 1. The performance index is then estimated by summing over the penalty coefficients corresponding to the finger indexes of the same hand used to hit two consecutive keys during the typing. Note that there is no penalty given to two consecutive keys hit by the fingers of different hands and the notation d in Table 1 specifies the distance between the two consecutive keys. It is noted that the home row usage is implicitly preferred in the keyboard arrangement because the penalty coefficient matrix penalizes the finger jumping over the home row by the distance (d) between the tapped keys.

Table 1
Unified penalty coefficient matrix for consecutive hitting fingers considering four ergonomic criteria.

1st finger	2nd finger				
	Thumb	Forefinger	Middle finger	Ring finger	Little finger
Thumb	$d+1$	2	2	2	2
Forefinger	1	$d+1$	7	10	8
Middle finger	1	6	$d+1$	11	9
Ring finger	1	9	10	$d+1$	12
Little finger	1	7	8	11	$d+1$

The penalty coefficients shown in Table 1 may be suitable for ordinary users with MFK typing. They are, however, not representative values in the SFK applications or for the users with physical disabilities. We propose to use a more general key-pair penalty coefficient matrix. Assume that the user has learned the typology with the keyboard arrangement k and the prediction algorithm p. Then there exists a mapping from the key index to the finger index used to hit the key. An $N \times N$ key-pair penalty coefficient matrix, denoted by $\tau_{kp}(u, v)$, is more complete to describe the physical comforts of finger usage for general users with arbitrary text entry devices. The coefficient values should be determined by consulting ergonomists and physicians. With the above practical consideration, our objective with the posture comforts for a keyboard arrangement k with the prediction algorithm p is defined as follows.

$$\text{Min } O_2(k) = \sum_{(u,v)\in D(\pi)} \tau_{kp}(u,v), \tag{2}$$

where $D(\pi)$ is the set containing any two consecutive keys that are hit to produce π. So the best keyboard arrangement is the one that minimizes the sum of the incurred penalty coefficient values $\tau_{kp}(u, v)$.

2.2. Disambiguation/prediction effectiveness

This category of objectives seeks to minimize the number of keystrokes used to disambiguate (in the MCK context) or to predict (in the general context) the intended word. Our prediction algorithm (as noted in Section 3.3) is a hybrid combining the advantages of both character-level and word-level prediction techniques. It predicts the most likely word based on any observed prefixes during the typing or disambiguates the candidate words when the user finishes typing the keys for the intended word. If the predicted word produced by the prediction algorithm matches the intended word, the user can accept the predicted word by pressing a reserved key. Otherwise, the user continues to type the next character. So the arrangement of the multiple characters on a key would significantly affect the effectiveness of the disambiguation/prediction algorithm. The effectiveness can be estimated in two-fold, the numbers of keystrokes and word clashes, as described as follows.

2.2.1. Keystroke

One of the most important objectives in designing the keyboard arrangement is the time consumed to enter a general text by the designed keyboard. The amount of time needed for typing depends upon many factors including the number of keystrokes, ergonomic criteria, learnability of the arrangement, the user's physical conditions, etc. The number of keystrokes is a direct (user-independent) consequence caused by the embedded prediction algorithm. Therefore, this objective is estimated by the total number of keystrokes needed to enter a text π using the given keyboard arrangement k with the prediction algorithm p. It is defined as follows.

$$\text{Min } O_3(k) = I^T \bullet l_{kp}(\pi), \tag{3}$$

where $l_{kp}(\pi)$ is the load distribution produced by typing π using keyboard arrangement k with the prediction algorithm p, I is a vector whose components are all equal to 1. So objective O_3 accounts for the number of keystrokes which is the sum of all the component values contained in $l_{kp}(\pi)$.

2.2.2. Word clash

Word clashes happen when ambiguity appears with multiple candidate words upon the time the user has finished tapping the corresponding keys of the intended word, i.e., when multiple candidate words have the same key sequence on a keyboard arrangement. Word clashes not only prolong the typing time, but also discourage the users. A good keyboard arrangement should minimize the number of word clashes. Let $O_4(k)$ denote the number of word clashes incurred by entering a text π using keyboard arrangement k with the prediction algorithm p, the objective seeks to minimize O_4 as much as possible.

2.3. Overall measure

The addressed GKAP problem involves four objectives and each objective has a respective range and unit of possible values. A proposal to the multi-objective optimization is to locate the Pareto front which plots the objective values of the solutions that are not dominated by any others. Solution x is said to dominate another solution y if x is strictly better than y according to at least one objective and x is no worse than y for the others. There exist some approaches for finding the Pareto front such as goal programming (Schniederjans, 1995) and epsilon programming (Yokoyama, 1996), however, most of them are very time-consuming in order to generate a well-represented front.

Another alternative is the weighting sum method (Lai and Li, 1999), i.e., transforming the multiple objectives into an overall objective by calculating their weighting sum. Here we adopt the weighting sum method operating on normalized objectives which can be obtained by dividing the objective values of the designed keyboard k_d with those of a reference keyboard k_r. A weighting overall measure serving as the objective function of the GKAP problem is thus proposed in the following.

$$\text{Min } O(k_d) = \sum_{i=1}^{4} w_i \times O_i(k_d)/O_i(k_r), \tag{4}$$

where w_i is the relative weight reflecting the contribution of objective O_i in the overall measure. The objective function can be easily converted to a 0–1 programming formulation which involves binary variables x_{ij} with respect to the designed keyboard k_d, where $x_{ij} = 1$ indicates the ith character is allocated on the jth key, and $x_{ij} = 0$ otherwise. A promising research direction has received much attention which strives to maximize the synergism between mathematical programming and metaheuristic computing (Glover and Hanafi, 2010). In this paper, we adopt the Cyber Swarm metaheuristic with the supplement of bounding programming technique to solve the GKAP problem.

3. Method

3.1. Review of particle swarm optimization and its variants

The particle swarm optimization (PSO) algorithm was first proposed by Kennedy and Eberhart (1995), and had exhibited a variety of successful applications, ranging from evolving weights and structure for artificial neural networks (Eberhart and Shi, 1998), manufacture end milling (Tandon, 2000), reactive power and voltage control (Yoshida et al., 1999), to image representation (Yin, 2004) and point pattern matching (Yin, 2006). The convergence and parameterization aspects of the PSO have also been discussed thoroughly (Clerc and Kennedy, 2002).

The PSO, inspired by the observations for bird flocking, simulates a form of swarm intelligence consisting of self-cognition and socio-cognition. The self-cognition is the awareness of personal best experience, denoted *pbest*, that has been recognized in its own historical behaviors. The socio-cognition is an influence between individuals (or particles, in the terminology of PSO) through direct/indirect interactions. The interaction is realized by affecting individual behaviors using the social norm that is accepted by an acceptance function. In the first proposal (Kennedy and Eberhart, 1995) the acceptance function is defined as the best experience observed by the entire swarm, denoted *gbest*. In the FIPS model (Mendes et al., 2004), it informs each individual of the set of best outcomes obtained by every other individual, rather than just the single best outcome found by the entire swarm. Yin et al. (2010) proposed the Cyber Swarm Algorithm which facilitates the reference set, a notion from scatter search (Laguna and Marti, 2003), to define the acceptance function by keeping a small common pool storing the most elite solutions from competitions among individuals. In essence, each particle (a trial solution) benefits from the best experience of its own and that of the swarm in various forms during the evolutionary process. It is evidenced from the literature that the performance of PSO strongly depends upon the form that it formulates the swarm intelligence. In this study, we adopted the cyber swarm algorithm which has been shown to be one of the most effective PSO variants.

3.2. Cyber Swarm algorithm for GKAP

This section articulates the specific features of the employed cyber swarm algorithm for tackling the general keyboard arrangement problem (GKAP).

3.2.1. Particle representation and fitness evaluation

Given a GKAP problem requiring an allocation of M characters to N keys, we can represent a candidate solution as a particle $\vec{P} = (p_1, p_2, \ldots, p_M)$ using a vector of M elements corresponding to the indices of the keys to which the M characters are to be allocated. Each element can have an integer value ranging between [1, N], indicating the index of allocated keys. The user may impose some constraints on the relationship between the element values according to the underlying problem type. When the SCK problem type ($M = N$) is considered, the particle is actually a permutation of M distinct numbers. On the other hand, when the MCK problem type ($M > N$) is taken into account, some elements would have the same value meaning that these characters will be allocated to the same key.

The fitness function evaluates the merit that a solution creates when solving an optimization problem. As noted in Section 2, we consider the quality of a solution to the GKAP by reference to two aspects: ergonomic fitness and disambiguation/prediction effectiveness. The overall measure $O(K_d)$ (defined in Eq. (4)) combining the two categories of factors plays a good role for the fitness function to be used to evaluate the fitness of a particle (a candidate solution) and it is adopted in our cyber swarm algorithm. As the overall measure is formulated as a minimization problem, the less the value of $O(K_d)$, the better the quality of the particle is.

It is observed that in many metaheuristic applications the evaluation of solution fitness is the most time-consuming part of the entire optimization process. It is more important to expedite the fitness evaluation in our problem because every instance of fitness evaluation involves a full scan of the training text which

usually contains thousands of words. Hence, we apply a bounding technique for fitness evaluation to save the computation time. Given a particle $\overrightarrow{P}$, let the fitness function of its current position and best experience position be $O(\overrightarrow{P})$ and $O(pbest_{\overrightarrow{P}})$, respectively. According to the PSO optimization procedure, the fitness of each individual is used for possible update of experience memory (*pbest* and *gbest*) instead of instructing the particle update or the selection for survival in the next generation. As our fitness function is a monotonically increasing function and we are aiming to minimize the fitness value, we can terminate the fitness evaluation for $O(\overrightarrow{P})$ once the intermediate value during the computation is greater than $O(pbest_{\overrightarrow{P}})$ stored in the memory, indicating the truth $O(\overrightarrow{P}) > O(pbest_{\overrightarrow{P}})$ even we do not finish computing the exact value of $O(\overrightarrow{P})$. Thus, we can use $O(pbest_{\overrightarrow{P}})$ as the upper bound for constraining the evaluation of $O(\overrightarrow{P})$. Once the intermediate value during the fitness computation for a particular particle exceeds the fitness value of its best experience observed so far, the calculation can be terminated and proceed directly to the fitness evaluation of the next particle. This expedition effect would be more profound in the later evolutionary phase because the quality of the upper bounds is constantly improved.

3.2.2. Learning from reference set

The standard PSO stipulates each particle should interact with its previous best solution and the best solutions of its neighbors, impairing the searching capability of the particles in complex objective space. The cyber swarm algorithm (Yin et al., 2010), on the other hand, considers a small collection of the best solutions observed overall by the entire swarm, making use of the reference set notion from the scatter search (Laguna and Marti, 2003). The specific features of cyber swarm algorithm are articulated as follows.

Each particle learns from interactions with members of the reference set, which is an array of historical best solutions, denoted *RefSol*[*i*], $i = 1,\ldots,R$, where R is the size of the reference set. The solutions *RefSol*[*i*] are sorted according to the decreasing order of their solution quality. In our problem setting which involves minimization of an objective function, the reference solutions are sorted in an increasing order of fitness values. Yin et al. (2010) have strategically selected three guiding points, namely *pbest*, *RefSol*[1] (or *gbest* using standard PSO terminology), and another reference solution *RefSol*[m], $m > 1$, to guide the search trajectories of each particle. Their experimental results have shown that the addition of RefSol[m] in guiding succeeds in significantly improving overall performance. To implement this strategy, the cyber swarm algorithm updates the velocity $(\overrightarrow{V}_i)$ in the jth dimension for the ith particle $(\overrightarrow{P}_i)$ as follows,

$$v_{ij}^{m} \leftarrow K\left(v_{ij} + (\varphi_1+\varphi_2+\varphi_3) \times \left(\frac{\omega_1\varphi_1 pbest_{ij} + \omega_2\varphi_2 RefSol[1]_j + \omega_3\varphi_3 RefSol[m]_j}{\omega_1\varphi_1+\omega_2\varphi_2+\omega_3\varphi_3} - p_{ij}\right)\right) \quad (5)$$

where K is the constriction factor, ϕ_k and ω_k are the cognitive coefficient and the relevance weight, respectively, for the kth guiding solution. The cognitive coefficient ϕ_k should be constrained in a range that is appropriate for the constriction factor K to ensure the convergence of the algorithm, Mendes et al. (2004) empirically suggested to use

$$\varphi_k \in U\left[0, \frac{4.1}{3}\right] \text{ for } K = 0.729. \quad (6)$$

In Yin et al. (2010), the results revealed that the fitness weighting strategy which determines the value of ω_k according to the guiding solution's fitness gives the best overall performance of the cyber swarm algorithm.

3.2.3. Dynamic neighborhood and diversification strategy

The standard PSO uses a *static* social network requesting that each particle always interacts with the same neighbors connected in the fixed neighborhood topology, while the cyber swarm algorithm employs a dynamic neighborhood which enlarges the interaction context by eliciting multiple viewpoints from the group the particle communicates with. Each such viewpoint is provided by an interaction with another member in the reference set, and the particle finally benefits from the influence of the best of all these interactions, allowing the particle to dynamically determine the best neighborhood topology at each move.

Diversification is an important search strategy which drives the course into uncharted regions and generates solutions that differ in significant ways from those seen before (Glover and Laguna, 1997). The cyber swarm algorithm adopts two diversification strategies. The minimum diversity strategy stipulates that any two members in the reference set should be separated from each other by a distance that is no less than a minimum threshold. The exploratory diversity strategy explores uncharted regions by linking under-explored regions to the overall best solution using the path relinking technique (Glover, 1998). Empirical study (Yin et al., 2010) has shown that the two diversity strategies can guide the search towards new promising and under-explored regions when the search gets stuck in a local optimum.

3.3. Prediction algorithm

As previously noted, a prediction algorithm is usually embedded in an MCK to disambiguate the intended word. Here, we propose to apply the prediction algorithm not only to tackle the word disambiguation problem, but also to reduce the number of keystrokes in the general context. Our prediction algorithm is a hybrid combining

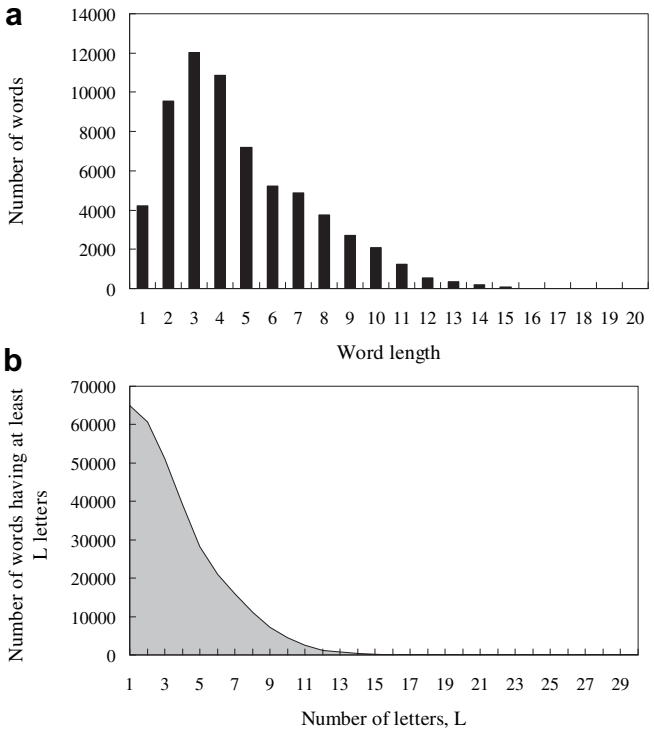

Fig. 1. Word length analysis.

Table 2
The sources of the text dataset.

sources	Word count	Percentages
Academic articles	24,331	28
Times magazine	10,533	12
News articles	10,317	12
Reader's digest	32,937	37
Fictions	9574	11
Total	87,692	100

the advantages of both character-level and word-level prediction techniques. Based on the tapped keys, our prediction algorithm presents the candidate word that is the most probable one according to a word usage frequency analysis. In other words, every time the user taps a key, the word with the highest frequency that contains any prefixes produced from the tapped keys is presented as the candidate word. Note that a sequence of tapped keys would generate a unique prefix or multiple prefixes in the SCK or the MCK cases, respectively. The user can either accept the predicted word by tapping a reserved key or continue to enter the next character on the keyboard.

To save the storage and computation complexity incurred by the word frequency analysis, the prediction algorithm only activates its function when the user has tapped at least four keys for producing the intended word. This practical strategy is based on two facts revealed from a word length analysis conducted on our text dataset which contains 87,692 words from diverse sources. The analysis discloses that the number of 3-letter words is the greatest among all (Fig. 1(a)), and that the number of words having at least L letters decreases dramatically when $L > 3$ (Fig. 1(b)). So the word prediction accuracy based on a prefix consisting of less than four letters would be low and the incurred storage and computation complexity are relatively high. It is more practical to execute the prediction algorithm only when the user has tapped a sufficient number of keys for the intended word and we set this number to four in the implementation.

4. Results and discussion

In this section, we introduce the dataset which is used to execute all the computer simulations in this study. The performance of the proposed method is compared with those of benchmark keyboards and other competing algorithms. A preliminary step for keyboard shape design using the proposed method is illustrated. To realize how learnable the produced layouts are, an empirical experiment involving humans is conducted.

4.1. Text dataset

In order to obtain a keyboard arrangement that has high quality performance over varying context, a text dataset with diverse contents should be created such that the text dataset well represents the general context. The sources of our text dataset contain academic articles, times magazine, news articles, reader's digest, and fictions. The number of words collected from these sources and its percentage to the dataset size is listed in Table 2. Overall, the text dataset has 87,692 words and each text source contributes 11–37% of the dataset body. We uniformly sample 75% (65,781 words) from the dataset as being the training set, and the rest of the dataset (21,911 words) as being the testing set. All the benchmark keyboards and the competing methods studied in this paper were evaluated using the same training set and testing set.

4.2. Benchmark keyboards and competing methods

The performance of the proposed method is verified by competing with benchmark keyboards and existing algorithms over the test dataset. All the competitors are introduced as follows.

4.2.1. The 3×3 benchmark keyboards

- *Alphabetic keyboard*. The Alphabetic keyboard design arranges the 26 English letters on the keys from left to right and from top to bottom. Alphabetic keyboard has been considered as the baseline design for performance comparison. In this paper, we consider the 3×3 Alphabetic keyboard arrangement as shown in Fig. 2(a).
- *Levine keyboard*. To maximize the expected accuracy of the character-level prediction algorithm, Levine et al. (1987) mathematically solved the optimization problem and proposed a keyboard arrangement as shown in Fig. 2(b).
- *TOC keyboard*. Foulds et al. (1987) switched the positions of three characters "t", "c", and "o" from the Alphabetic keyboard and produced the so called "TOC" keyboard as shown in Fig. 2(c). TOC keyboard has been empirically shown to be one of the most effective keyboards which require a less number of keystrokes when typing a text with an n-gram prediction algorithm.
- *Frequency keyboard*. In order to balance the tapping load on each key, Arnott and Javed (1992) proposed a Frequency keyboard which arranges the characters such that the frequency that each key will be struck is as close as possible. The arrangement of Frequency keyboard is shown in Fig. 2(d).

4.2.2. The 10×3 benchmark keyboards

- *QWERTY keyboard*. The QWERTY keyboard is the most widely used commercial keyboard and we use it as a reference keyboard for performance evaluation, though the QWERTY keyboard pays no attention to any ergonomic criteria.
- *Dvorak keyboard*. The Dvorak keyboard is the first keyboard design taking ergonomic criteria into consideration (Cassingham, 1986), including home row usage, co-occurrence frequency of consecutive letters in English, and easiness of typing learning.

4.2.3. Competing algorithms

- *Random keyboard*. The letters are arbitrarily arranged on available keys. After all the 26 letters have been allocated, the fitness of the random keyboard is evaluated. Given an allowable number of fitness evaluations, the random keyboard

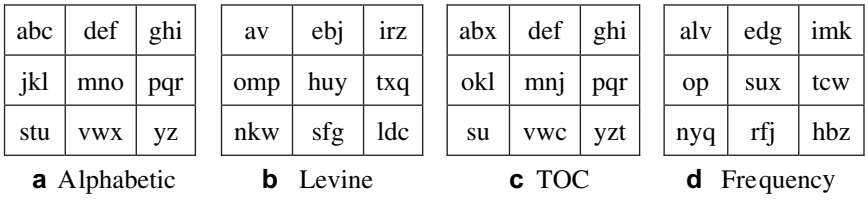

Fig. 2. The 3×3 arrangement of benchmark keyboards.

Table 3
The respective objective values and the overall score obtained by the competing 3 × 3 keyboards.

Objectives	Alphabetic	Levine	TOC	Frequency	Cyber w/o p	Cyber w. p
O_1	0.03506	0.00854	0.00996	0.010165	0.001372	0.00149
O_2	510,294	510,294	510,294	510,294	510,294	394,308
O_3	106,927	106,927	106,927	106,927	106,927	90,699
O_4	7546	5567	4655	5509	2315	1748
O	1.0	0.7453	0.7252	0.7550	0.5865	0.4738

arrangement can be repeatedly tried (however, these trial arrangements are independent of each other) and the one with the minimum fitness among these trials is considered as the final arrangement of the random keyboard.

- *Heuristic keyboard*. Starting with an arbitrary random keyboard arrangement, a 2-swap heuristic is iteratively applied to improve the fitness of the trial keyboard. The 2-swap heuristic is performed by randomly selecting two letters which are currently located on different keys and exchange the positions of the two letters. Each time a 2-swap heuristic is applied, we re-evaluate the fitness of the new keyboard arrangement. If the fitness is improved (with a decrement on the value) we consider the new arrangement as the current trial for performing the next 2-swap operation; otherwise, the previous arrangement is resumed. With the allowable number of fitness evaluations, the trial keyboard with the minimum fitness observed so far is considered as the final arrangement of the heuristic keyboard.
- *Standard PSO keyboard*. By applying the constriction Type-1 PSO algorithm (Clerc and Kennedy, 2002) (which is one of the most widely used version and considered as the standard form of PSO), we can obtain a PSO-optimized keyboard arrangement by consuming the allowable number of fitness evaluations. The best keyboard arrangement with the minimum fitness observed by any particles in the swarm is considered as the final arrangement of the Standard PSO keyboard.
- *Cyber Swarm keyboard*. This keyboard arrangement is obtained by performing the specific cyber swarm algorithm and the prediction algorithm presented in Section 3, using the same number of allowable fitness evaluations.

4.3. Comparative performances

4.3.1. Comparison with benchmark keyboards

There are several benchmark keyboards proposed in the commercial market or literature. Each benchmark keyboard has its own motivations and design purposes. Specifically, we compare the performance of the proposed method with those of 3 × 3 and 10 × 3 benchmark keyboards because they are commonly used in hand-held devices and standard text entry devices, respectively. In all the experiments, we obtain the Cyber Swarm keyboard by allowing 4000 fitness evaluations on the training set containing 65,781 words. All the competing keyboards are evaluated using the testing set consisting of 21,911 words.

4.3.1.1. The 3 × 3 benchmark MCK keyboards. We compare the performances among Alphabetic, Levine, TOC, Frequency, Cyber Swarm without performing the prediction algorithm (Cyber w/o p), and Cyber Swarm embedding the prediction algorithm (Cyber w. p). As the 3 × 3 keyboards are usually implemented in a small area for mobile use and the typing is usually performed with a single finger or a stylus device, the ergonomic penalty coefficients are set to the same value. Table 3 shows the respective objective values and the overall score obtained by the competing keyboards. We see that the Cyber Swarm keyboard (either with or without the implementation of the prediction algorithm) significantly outperforms the other competing keyboards according to the overall measure (O value). It is noted that TOC keyboard surpasses Alphabetic, Levine, and Frequency keyboards and the Alphabetic keyboard is the worst arrangement among all. Levine keyboard is slightly better than Frequency keyboard. We can also observe similar results by reference to the keystroke efficiency measure defined in Lesher et al. (1998) which compared the performance of Alphabetic, Levine, TOC, and Frequency keyboards. The keystroke efficiency measure is reflected in one of our objectives: the number of word clashes (O_4). As listed in Table 3, the Cyber w. p keyboard encounters 1748 word clashes for typing the testing text with 21,911 words, while the Cyber w/o p keyboard generates 2315 word clashes. Both of our Cyber Swarm keyboards outperform the other benchmark keyboards, revealing that our approach is indeed general and the produced keyboard is better than the existing keyboards according to their original performance index. Fig. 3 shows the layout of the 3 × 3 Cyber w. p keyboard.

4.3.1.2. The 10 × 3 benchmark SCK keyboards. We compare the performance of the Cyber Swarm keyboard with that of QWERTY and Dvorak keyboards. In order to produce a compact keyboard in a 10 × 3 layout, both QWERTY and Dvorak keyboards place four marks (semicolon, comma, period, and forward-slash) on corner keys such that the 26 letters can be intensely arranged. For a fair comparison, we also place semicolon, comma and period marks in the leftmost positions of the upper row and the forward-slash mark in the rightmost position of the lower row. So the problem is reduced to an SCK keyboard arrangement problem and we are aiming to arrange 26 English letters on 26 available keys. To deal with the keyboard arrangement for common users, the tapping load distribution and the ergonomic penalty coefficients (http://intelligence.im.ncnu.edu.tw/coefficients.htm) transformed from the numerals provided in Eggers et al. (2003) are employed. It is seen from Table 4 that the Cyber w. p and the Cyber w/o p keyboards obtain the best and the third best overall scores (O value), respectively, Dvorak keyboard is ranked in the second place, and QWERTY keyboard is the worst arrangement as expected. Dvorak keyboard produces the minimal value for key accessibility (O_1) by taking into account the home row usage. In the light of Eggers' ideal tapping load distribution where 43.5% of the key strokes are done in the home row and the penalty coefficient matrix which penalizes the finger jumping over the home row by the distance (d) between the tapped keys, the Cyber w. p and the Cyber w/o p keyboards are also able to arrange frequently used

awz	lo	be
hn	jqtv	ciy
dfp	kmrx	gsu

Fig. 3. The 3 × 3 arrangement of the Cyber Swarm keyboard.

Table 4
The respective objective values and the overall score obtained by the competing 10×3 keyboards.

Objectives	QWERTY	Dvorak	Cyber w/o p	Cyber w. p
O_1	0.04789	0.01452	0.02411	0.02088
O_2	500,238	469,948	420,447	391,452
O_3	106,927	106,927	106,927	89,889
O_4	0	0	0	0
O	1.0	0.5606	0.5860	0.5148

letters in the home row. As seen in Table 4, the Cyber w. p and the Cyber w/o p keyboards both outperform QWERTY keyboard by reference to the value of O_1. In fact, if we compare the arrangement of Dvorak keyboard with that of Cyber w. p keyboard (as shown in Fig. 4), 9 out of the 10 letters placed in the home row are the same for the two keyboard arrangements but the letters are arranged in different orders.

The Cyber w. p and the Cyber w/o p keyboards produce lower values for posture comfort (O_2) than the other competing keyboards. This manifests the effectiveness of the employed optimization method (Cyber Swarm Algorithm) aiming to minimize the sum of the penalty coefficients incurred by the keyboard arrangement. The Cyber w. p keyboard generates the minimal number of keystrokes (O_3) because it performs the prediction algorithm which greatly alleviates the need for typing all letters of the intended word. Moreover, as the experiment is conducted in the SCK context, no word clashes (O_4) are generated by any tested keyboards.

Eggers et al. (2003) evaluated the performance of their designed keyboard and obtained a global score of 0.593 using the QWERTY keyboard as a reference (see Eq. (4)). They used the load and accessibility, hand alternation, consecutive usage of the same finger, avoid big step, and hit direction as the performance indices, these measures have been combined in two of our objectives: key accessibility (O_1) and posture comfort (O_2). Hence, the global scores for the proposed Cyber Swarm keyboards can be easily computed using the numerical values listed in Table 4. In particular, the Cyber w. p and the Cyber w/o p keyboards obtain the global scores of 0.609 and 0.672, respectively, both of which are slightly inferior to the global score (0.593) of Eggers' keyboard. This is because our Cyber Swarm keyboards are designed using the proposed general keyboard arrangement problem (GKAP) model and the keyboard arrangement is optimized according to the disambiguation effectiveness measures (O_3 and O_4), in addition to the ergonomic criteria (O_1 and O_2) adopted by Eggers et al. (2003).

4.3.2. Comparison with competing algorithms

In order to evaluate the performance of the Cyber Swarm keyboard in a general context, we further conduct the experiments for three different ergonomic penalty coefficient matrices generated at random for 3×3 and 8×2 keyboard layouts, respectively. These random matrices and different keyboard layouts can represent the varying cases for the required ergonomic constraints according to the physical conditions of the particular user. We compare the performance of the Cyber Swarm keyboard (embedding the prediction algorithm) with those of Random, Heuristic, and Standard PSO keyboards. 10 independent runs for each algorithm are conducted and the overall score for each ergonomic

;	,	.	G	F	J	W	R	Z	U
A	N	E	S	H	I	T	O	D	Q
M	P	C	K	Y	L	B	X	V	/

Fig. 4. The 10×3 arrangement of the Cyber Swarm keyboard.

Table 5
Overall score for each ergonomic penalty matrix and the average over these scores in the 3×3 keyboard arrangement.

Objectives	Random	Heuristic	Standard PSO	Cyber Swarm
Matrix 1	0.711616	0.612863	0.520845	0.476498
Matrix 2	0.720193	0.615327	0.544547	0.487513
Matrix 3	0.693232	0.611126	0.525085	0.485108
Average	0.708347	0.613105	0.530159	0.483040

penalty matrix and the average over these scores are reported. Tables 5 and 6 list the results for 3×3 and 8×2 keyboard layouts, respectively. It is seen that in all cases Cyber Swarm keyboard gives the best performance, Standard PSO keyboard ranks at the second place, Heuristic keyboard occupies the third place, and Random keyboard produces the worst arrangement.

Fig. 5 shows the best overall score that can be obtained by each competing algorithm as the number of function evaluation increases. The Heuristic method has the fastest improvement ratio on the best overall score during the first 30 function evaluations due to the greedy optimization procedure performed. However, it fails to constantly improve the best overall score after consuming more than 320 function evaluations. The Cyber Swarm method derives worse overall score during the first 100 function evaluations because it spends extra time to explore the solution space in order to build up a reference set containing elite solutions. After consuming 200 function evaluations, the Cyber Swarm method becomes the best one that gives the minimum overall score among all competing methods and constantly improves the score value as the number of function evaluations increases. On the other hand, the Standard PSO focuses more on intensification of local optimum found and turns out to be the second best method in the competition. Finally, the Random method conducts the search without using any guidance information, it is not surprising that the Random method gives the worst overall score in most of the time durations.

4.3.3. Customized keyboard shape design

As previously noted, our method for GKAP can be applied in multiple scenarios. One of the useful applications is dedicated to preliminary customized keyboard shape design for motor-impaired users with a stylus as the text entry device. Given M letters and N keys with $M < N$, our method would allocate the characters using only a subset of the available keys according to the specified objective weights. The experiment has different settings from that used in previous experiments. (1) For preliminary keyboard shape design with motor-impaired users, the Eggers' ideal tapping load distribution which is designed for common users should not be adopted. We have replaced the Eggers' ideal tapping load distribution by a uniform distribution, as such no prior information for significant keys is assumed. (2) As the preliminary keyboard shape design problem involves a greater number of available keys for allowing feasible keyboard shapes, a longer running time than that used in previous experiments should be allocated to the proposed Cyber Swarm method in order to produce a near-optimal result. In particular, we let the program run for 8000 fitness evaluations.

Table 6
Overall score for each ergonomic penalty matrix and the average over these scores in the 8×2 keyboard arrangement.

Objectives	Random	Heuristic	Stand PSO	Cyber Swarm
Matrix 1	0.605059	0.566905	0.450586	0.414563
Matrix 2	0.650105	0.620282	0.479017	0.447792
Matrix 3	0.662459	0.593952	0.464919	0.422915
Average	0.639208	0.593713	0.464841	0.428423

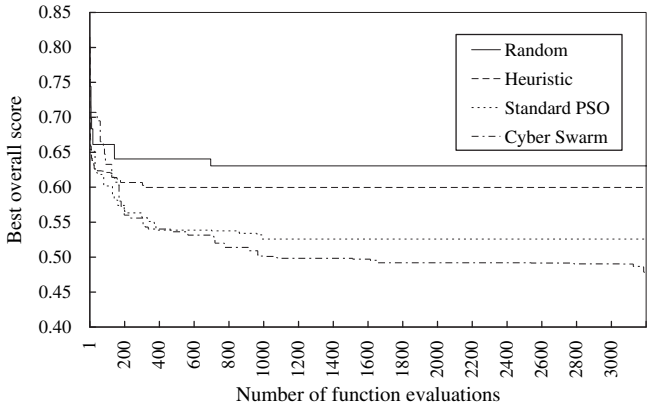

Fig. 5. The best overall score that can be obtained by each competing algorithm as the number of function evaluation increases.

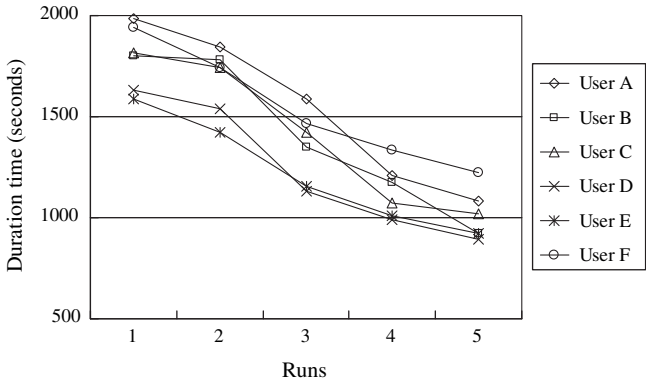

Fig. 7. Duration time of each run that the testers performed the typing.

In the following illustrative example, we assume the information regarding the physical disabilities of the motor-impaired user is known a priori and has been converted into the ergonomic penalty coefficient matrix $\tau_{kp}(u, v)$. To emphasize the objective importance of the posture comfort for the motor-impaired user, we set the weight corresponding to this objective to 0.7 and the total weight for the other three objectives to 0.3. An initial keyboard with 36 keys arranged in 12×3 layout is used for allocating the 26 English letters. As shown in Fig. 6, the penalty coefficient values are generated at random with respect to the range $(0, 3)$ for consecutive hit keys in the shaded area and to the range $(9, 12)$ for the other key pairs. This is to visualize the result of the keyboard shape design if most of the characters are arranged in the shaded keys with lower penalty coefficient values. Fig. 6 shows the keyboard arrangement result obtained by our Cyber Swarm method. It is seen that most of the letters are intensely arranged in the middle shaded area which is given lower ergonomic penalties. As letter "Z" has the least use frequency among all English letters, it is not placed in the shaded area due to the tradeoff that it would incur a larger number of keystrokes for typing a text if "Z" is arranged in the same key with other letters. This experiment discloses the effectiveness of the proposed method in serving as a preliminary process for customized keyboard shape design for motor-impaired users.

4.3.4. Computation efficiency

It is important to expedite the fitness evaluation process in our addressed problem because every instance of a complete function evaluation involves a full scan of the training text which usually contains thousands of words (it contains 65,781 words in our current case). Hence, we apply a bounding technique for intelligent fitness calculation as noted in Section 3.2.1 to save the computation time. With the observation from our experimental results, the employed bounding technique can reduce about 71% of the needed computation time. It typically decreases the computation time from 14 h to 4 h.

4.3.5. Empirical human–computer interaction experiment

To validate the proposed method on a large dataset containing 87,692 words, the previous experiments were conducted using computer simulations. In this subsection, we employ an empirical approach to evaluate the performance regarding human–computer interaction (HCI) issues such as the effectiveness of disambiguation and the learnability of the keyboard layout. Six persons (http://intelligence.im.ncnu.edu.tw/) who have at least 4-year experience in QWERTY keyboard typing for technical articles were invited to join this experiment. Each person chose an article with less than 500 words and typed the article using our 10×3 Cyber Swarm keyboard, which is implemented by pasting the Cyber Swarm keyboard arrangement (see Fig. 4) on the QWERTY keyboard and switching the keyboard interrupt calls from QWERTY keys to Cyber Swarm keys. To analyze the effectiveness of HCI design for the Cyber Swarm keyboard, we let each person type his/her selected article five times with 20 min of resting between two repetitions. After the first run of article typing, the testers have comments among the following. "It is frustrating that I have to find where every letter is located on the keyboard", "It's interesting that I don't have to finish the complete typing of every word, which is usually correctly guessed and shown on the screen when I typed the prefix containing 5 or 6 letters", "I cannot focus simultaneously on finding the letters on the keyboard and the context of the article printed on the paper". It seems that the users have difficulties in using the new keyboard arrangement efficiently, though the new product's function is interesting. When all the users have performed the second try of typing, they gave similar comments as previously noted, but some promising results like "the prediction algorithm could be useful in saving time" were observed. At last, when the users have finished five repetitions for typing the article, we received the comments: "I have known the locations of most of the letters", "Sometimes I am still puzzled due to my prior experiences with QWERTY keyboard", "The prediction algorithm is useful especially when I am more familiar with the guessing, so I can proceed to the typing of the next word without hesitating", "I feel my last run of typing is significantly more efficient than my first run of experiment". Overall, we receive positive feedbacks from the testers after the five repetitive runs. Moreover, we recorded the duration time of each run that the testers performed the typing because the learning time is one of the typical measures for understanding the usage learnability of a new keyboard layout. As shown in Fig. 7, the testers are not performing efficient typing at the first run but gradually improve their familiarity with the layout and the prediction

			Z	HLQT	BX	E	S				
			N	JW	MOU	AG	IP	FRY			
				C	V	D	K				

Fig. 6. Preliminary keyboard shape design for motor-impaired users.

algorithm so the duration time is shortened at subsequent runs. The average typing rate for all testers is 16.7 words/min at the first run, and it increases to 29.7 words/min at the fifth run. This indicates that the Cyber Swarm keyboard can be learned quickly and effectively by traditional QWERTY users, the typing rate almost doubles after a reasonable amount of practice.

5. Conclusions

In this paper we have proposed a Cyber Swarm Algorithm-based keyboard arrangement to be employed in varying context. Traditional keyboard arrangement algorithms aim at finding an optimal solution to a specific objective such as the design of Dvorak keyboard is intended to meet the ergonomic figures of merit in the SCK application and Levine keyboard is approximately optimal in reference to producing minimal character-level ambiguities in the MCK context. We anticipate that a keyboard arrangement considering multiple objectives can create additional benefits that cannot be obtained by traditional methods. Our Cyber Swarm method accommodates ergonomic criteria and disambiguation/prediction effectiveness simultaneously. Experimental results manifest that the Cyber Swarm keyboard outperforms several benchmark keyboards and other competing algorithms. We have also shown an illustrative example for preliminary keyboard shape design which could be very useful in customized keyboard production for motor-impaired users whose physical capacity has been evaluated a priori. Finally, an empirical experiment involving human subjects is performed in order to evaluate the human–computer interface performance such as the prediction effectiveness and learnability of the new keyboard layout.

References

Arnott, J.L., Javed, M.Y., 1992. Probabilistic character disambiguation for reduced keyboards using small text samples. Augmentative and Alternative Communication 8, 215–223.

Cassingham, R.C., 1986. The Dvorak Keyboard: the Ergonomically Designed Typewriter, Now An American Standard. Freelance Communications, Arcata, California.

Clerc, M., Kennedy, J., 2002. The particle swarm explosion, stability, and convergence in a multidimensional complex space. IEEE Transaction on Evolutionary Computation 6, 58–73.

Deshwal, P.S., Deb, K., 2003. Design of an optimal hindi keyboard for convenient and efficient use. Technical Report KanGAL Report No. 2003005. Indian Institute of Technology, Kanpur.

Eberhart, R.C., Shi, Y., 1998. Evolving artificial neural networks. In: Proceedings of the International Conference on Neural Networks and Brain, PL5–PL13.

Eggers, J., Feillet, D., Kehl, S., Wagner, M.O., Yannou, B., 2003. Optimization of the keyboard arrangement problem using an ant colony algorithm. European Journal of Operational Research 148 (3), 672–686.

Fitts, P.M., 1992. The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental Psychology 121 (3), 262–269.

Foulds, R.A., Soede, M., van Balkom, H., 1987. Statistical disambiguation of multi-character keys applied to reduce motor requirements for augmentative and alternative communication. Augmentative and Alternative Communication 3, 192–195.

Glover, F., Laguna, M., 1997. Tabu Search. Kluwer Academic Publishers, Boston.

Glover, F., 1998. A template for scatter search and path relinking. Artificial evolution. Lecture Notes in Computer Science 1363, 13–54.

Glover, F., Hanafi, S., 2010. Metaheuristic search with inequalities and target objectives for mixed binary optimization – part I: exploiting proximity. International Journal of Applied Metaheuristic Computing 1 (1), 1–15.

Kennedy, J., Eberhart, R.C., 1995. Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, IV, 1942–1948.

Kwon, S., Lee, D., Chung, M.K., 2009. Effect of key size and activation area on the performance of a regional error correction method in a touch-screen QWERTY keyboard. International Journal of Industrial Ergonomics 39 (5), 888–893.

Ladany, S.P., 1975. A model for optimal design of keyboards. Computers and Operations Research 2 (1), 55–59.

Laguna, M., Marti, R., 2003. Scatter Search: Methodology and Implementation in C. Kluwer Academic Publishers, London.

Lai, K.K., Li, L., 1999. A dynamic approach to multiple-objective resource allocation problem. European Journal of Operational Research 117, 293–309.

Lee, Y.L., 2010. Comparison of the conventional point-based and a proposed finger probe-based touch screen interaction techniques in a target selection task. International Journal of Industrial Ergonomics 40 (6), 655–662.

Lesher, G.W., Moulton, B.J., Higginbotham, D.J., 1998. Optimal character arrangements for ambiguous keyboards. IEEE Transactions on Rehabilitation Engineering 6, 415–423.

Levine, S.H., Goodenough-Trepagnier, C., Getschow, C.O., Minneman, S.L., 1987. Multi-character key text entry using computer disambiguation. In: Proceedings of the Tenth Annual Conference on Rehabilitation Engineering. RESNA, Washington, DC, pp. 177–179.

Li, Y., Chen, L., Goonetilleke, R.S., 2006. A heuristic-based approach to optimize keyboard design for single-finger keying applications. International Journal of Industrial Ergonomics 36, 695–704.

Light, L.W., Anderson, P.G., 1993. Typewriter Keyboards via Simulated Annealing. AI Expert.

Mackenzie, I.S., Soukoreff, R.W., 2002. Text entry for mobile computing: models and methods, theory and practice. Human-Computer Interaction 17, 147–198.

Mendes, R., Kennedy, J., Neves, J., 2004. The fully informed particle swarm: simpler, maybe better. IEEE Transaction on Evolutionary Computation 8, 204–210.

Minneman, S.L., 1985. A simplified touch-tone telecommunications aid for deaf and hearing impaired individuals. In: Proceedings of the Eighth Annual Conference on Rehabilitation Technology. RESNA, Washington, DC, pp. 209–211.

Oommen, B.J., Valiveti, R.S., Zgierski, J.R., 1989. Application of Genetic Algorithms to the Keyboard Optimization Problem. Technical Report. Carleton University, Ottawa, Canada.

Oommen, B.J., Valiveti, R.S., Zgierski, J.R., 1991. An adaptive learning solution to the keyboard optimization problem. IEEE Transactions on Systems, Man, Cybernetics 21 (6).

Park, Y.S., Han, S.H., 2010a. Touch key design for one-handed thumb interaction with a mobile phone: effects of touch key size and touch key location. International Journal of Industrial Ergonomics 40 (1), 68–76.

Park, Y.S., Han, S.H., 2010b. One-handed thumb interaction of mobile devices from the input accuracy perspective. International Journal of Industrial Ergonomics 40 (6), 746–756.

Schniederjans, M.J., 1995. Goal Programming: Methodology and Applications. Kluwer Academic Publishers, Norwell, USA.

Sorensen, K., 2007. Multi-objective optimization of mobile phone keymaps for typing messages using a word list. European Journal of Operational Research 179 (3), 838–846.

Tanaka-Ishii, K., Inutsuka, Y., Takeichi, M., 2002. Entering text with a four-button device. In: Proceedings of the 19th International Conference on Computational Linguistics, vol. 1, pp. 1–7.

Tandon, V., 2000. Closing the Gap Between CAD/CAM and Optimized CNC End Milling. Master thesis, Purdue School of Engineering and Technology, Indiana University Purdue University, Indianapolis.

Tomatis, L., Nakaseko, M., Laubli, T., 2009. Co-activation and maximal EMG activity of forearm muscles during key tapping. International Journal of Industrial Ergonomics 39 (5), 749–755.

Wagner, M.O., Yannou, B., Kehl, S., Feillet, D., Eggers, J., 2003. Ergonomic modeling and optimization of the keyboard arrangement with an ant colony algorithm. Journal of Engineering Design 14 (2), 187–208.

Yin, P.Y., 2004. A discrete particle swarm algorithm for optimal polygonal approximation of digital curves. Journal of Visual Communication and Image Representation 15, 241–260.

Yin, P.Y., 2006. Particle swarm optimization for point pattern matching. Journal of Visual Communication and Image Representation 17 (1), 143–162.

Yin, P.Y., Glover, F., Laguna, M., Zhu, J.X., 2010. Cyber swarm algorithms – improving particle swarm optimization using adaptive memory strategies. European Journal of Operational Research 201 (2), 377–389.

Yokoyama, K., 1996. Epsilon approximate solutions for multiobjective programming problems. Journal of Mathematical Analysis and Applications 203 (1), 142–149.

Yoshida, H., Kawata, K., Fukuyama, Y., Nakanishi, Y., 1999. A particle swarm optimization for reactive power and voltage control considering voltage stability. In: Proceedings of the International Conference on Intelligent System Application to Power Systems, pp. 117–121.

Zhai, S., Hunter, M., Smith, B.A., 2002a. Performance optimization of virtual keyboards. Human-Computer Interaction 17, 229–269.

Zhai, S., Sue, A., Accot, J., 2002b. Movement model, hits distribution and learning in virtual keyboarding. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Changing Our world, Changing Ourselves, pp. 17–24.