2-408-79
2017/3/2

組み合わせ最適化手法の応用に関する研究 経路依存性を考慮したキーボード配列の最適化

髙木遼太[†1]　藤井慶[†1]

概要：今日，私たちが使っている多くのキーボードは QWERTY 配列である．しかしながら，このキー配列は元々タイプライターで用いられていたため，あえて打鍵速度を落とすように設計されているとも言われている．これまでにも代案のキー配列が考案されてきたが，そのどれもが一般には普及していない．その一因として経路依存性の問題が挙げられる．すなわち，多くの人は既に QWERTY 配列に慣れた状態にあり，そこから代替配列のタイピングを学習し直すということが難しいため，QWERTY 配列を続けることが考えられる．そこで本稿では，この経路依存性に起因する QWERTY 配列の問題に対して，組み合わせ最適化手法の応用によって解決を試みる．

Research on Application of Combinatorial Optimization Optimization of Keyboard Layout Considering Path Dependency

RYOTA TAKAKI[†1]　KEI FUJII[†1]

Abstract: Many people use QWERTY layout keyboard on VDT work. However, it is said that this keyboard layout reduces the speed of typing because it has been used in obsolete typewriters. Although alternative keyboard layouts such as the Dvorak simplified keyboard have been proposed, they don't become widespread. The path dependency is one of factors that masses continue using legacy keyboard layout. It is difficult to migrate to new layout for they get used to typing on old layout. This article describes trial to solve the problems of path dependency by applying combinatorial optimization.

1. 緒言

今日, 私たちが使っている多くのキーボードは QWERTY 配列である[1]．しかしながら，このキー配列は，元々タイプライターで用いられていたものであり，アームの衝突を防ぐために，あえて打鍵速度を落とすように設計されているとも言われている．この QWERTY 配列の代案として，Dvorak 配列や NICOLA 配列など，より入力効率を重視したキー配列が考案されてきたが，そのどれもが一般には普及していない．その一因として経路依存性の問題が考えられる．経路依存性とは，「あらゆる状況下において，人間やその組織の取りうる判断は，それまでの過去や歴史的な経緯によって制約を受ける」[2]という性質である．つまり，私たちの多くは, QWERTY 配列に慣れてしまっているが故に，他のキー配列に移行する際の負担が大きくなってしまい，非効率なキー配置の使用継続を余儀なくされているのである．

そこで本研究では，キーの配置を一度にまとめて変更するのではなく，少しずつ慣らしながら変えていくことで，経路依存性の問題を考慮しつつキー配列の最適化が行えるシステムを提案する．

†1 熊本高等専門学校
National Institute of Technology, Kumamoto College

2. 提案手法について

2.1 経路依存性を考慮したキー配列の最適化

本研究では，最適なキー配列を「タイピング時に手をホームポジションからできるだけ動かさずに済む配列」と定義する．言い換えれば，各指のホームポジションからの移動量が最小となるキー配列である．これを踏まえた上で，最適化の手順と最適性の評価方法について説明する．

最適化にはユーザの各キーの打鍵回数を用いる．ユーザのキー入力統計を用いることで, ユーザの使用言語, 業種, 癖などにも対応した最適化ができることも期待される．

今N個のキーを持つキーボードのa番目$(1 \le a \le N)$のキーをk_aとし，k_aが押された回数をc_a，k_aに最も近いホームポジション上のキーとk_aとの距離をd_aとおくと，k_aの配置に関する評価値e_aは次式で定義される．

$$e_a = \frac{c_a}{\sum_{i=1}^{N} c_i} \times d_a \quad (1)$$

これより，キー配列の全体の評価値Eは次式で定義される．

$$E = \sum_{i=1}^{N} e_i \quad (2)$$

この評価値Eが最小となるのが最適なキー配列である．つまり，最適なキー配列を見つけることは，目的関数Eが最小となるキーの並びを求める組み合わせ最適化問題を解くことと同義である．

本研究では，この最小化問題に対して貪欲法を用いたアプローチを試みる．今，i番目とj番目のキーの入れ替えを考える．押された回数c_iとc_jを入れ替えて(1)式を再計算した場合の各々のキーの評価値をe'_iとe'_jとすれば，このキーのペア評価値$e_{i,j}$は，

$$e_{i,j} = (e'_i + e'_j) - (e_i + e_j) \qquad (3)$$

と定義される．貪欲法では，$e_{i,j}$を最小にするペアが目的関数Eを最小値にもっとも近づける一手として選択される．そしてこの基準に基づきキーを繰り返し入れ替えていくことで，Eを最小に近づけていく．

貪欲法は反復の初期段階において評価値を大きく減少させる逐次的な最適化手法である．キー配列最適化の問題には経路依存性があるため，一気に最適化してしまうのではなく逐次的に最適解に近づくアプローチを採ることで，ユーザが少しずつ新しい配列に慣れていくことが期待される．またユーザによっては最適解に至るまでキー入れ替えを繰り返さず数個の入れ替えのみを行う可能性がある．この点においても貪欲法は最初の数回で大きく評価値を下げることが期待される．

2.2 システムの全体概要

本研究における提案手法はハードウェアキーボード，ソフトウェアキーボードのいずれでも実現可能だが今回はソフトウェアキーボードを用いる．ソフトウェアキーボードを用いる理由として，まずキーがグラフィックスで表現されるため，位置や形状の変更が容易なことが挙げられる．すなわち，ハードウェアキーボードではキーを入れ替えるとキー刻印と入力文字との間に齟齬が生じてしまうが，ソフトウェアキーボードではこの問題を回避できる．またユーザの手の大きさに沿ってキーボードサイズを変えるような柔軟性を持たせることもできる．さらにソフトウェアキーボードでは QWERTY 配列とフリック入力とを組み合わせる手法[3]も提案されており，入力の効率化が期待される．

本研究で開発したキーボードを図 1 に示す．後述する評価実験のため，今回は Enter キーや Shift キーなど，通常は正方形ではないキーの形状も他のキーと統一していることに注意されたい．

図 1　キー配置を変更可能なソフトウェアキーボード

2.3 キー配置の最適化

キー配列の最適化を行うため，本ソフトウェアキーボードには以下の 2 つの特殊キーを追加した．

(1) 可視化キー

可視化キーを押すと，キーの入力頻度分布を図 2 のように可視化する．多く押されているキーほど濃い赤色で表示する．

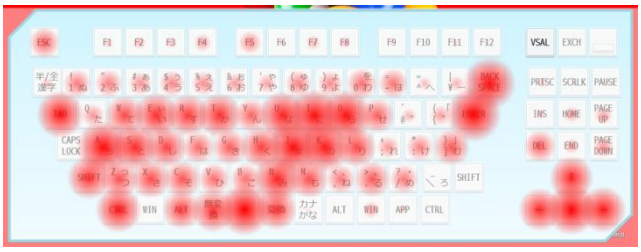

図 2　可視化キー実行例

(2) 最適化キー

最適化キーを押すと，図 3 のようなウィンドウが表示される．1 行あたり 1 対のキー入れ替え候補が並んでおり，所望の行の Exchange ボタンを押すと 2 つのキーが入れ替わる．中央の列は(3)式によるキーのペアの評価値であり，一番上の Total:の右側の数値がキー配列の評価値，すなわち(2)式による評価関数Eの値である．Exchange ボタンが押される度に，すべてのキーのペアの評価値が再計算され，キー入れ替えの候補リストも更新される．この操作を繰り返していくことで，キー配列を最適なものに近づけていく．

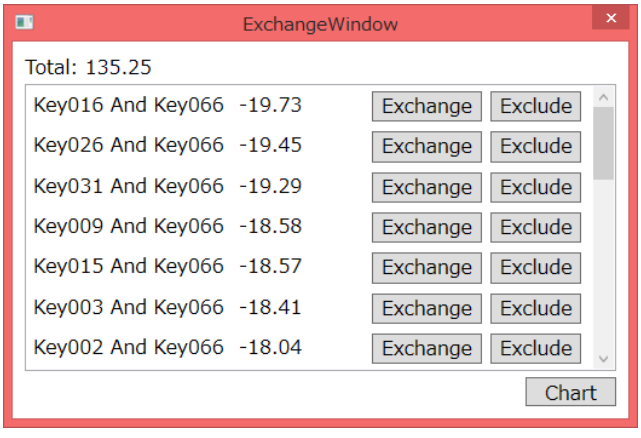

図 3　最適化ウィンドウ

2.4 その他の機能

本ソフトウェアキーボードには，先述の機能以外にもユーザビリティを向上させるための独自の機能を追加している．

2.4.1 手の位置に合わせた表示機能

ソフトウェアキーボードを表示する際，表示場所を手の

位置に合わせることで入力速度を向上できる[4]ことが知られている．本研究のソフトウェアキーボードでは，キーボードが非表示の時，スクリーンをロングタップして画面全体を白いレイヤーで覆った上で，スクリーンを 5 本の指でマルチタップすると，あらかじめ登録しておいたホームポジションを基に，手の位置にキーボードが表示される．通常のソフトウェアキーボードのように，タスクトレイ中のアイコンをタップすることでも表示できる．例を図 4 に示す．

図 4　手の位置に合わせて表示（黒丸が指の位置）

2.4.2 パワーポイントを用いたレイアウト編集機能

本ソフトウェアキーボードは，JSON 形式のレイアウトファイルを編集することで，自由にキー配置を変更することができる．しかしながら，このファイルを直接書き換えるのは少々敷居が高い．そこで，図 5 のように，パワーポイント上の図形としてキーボードレイアウトをデザインし，そのスライドを付属のツールで JSON 形式に変換すれば，そのまま実際に使用することができる．

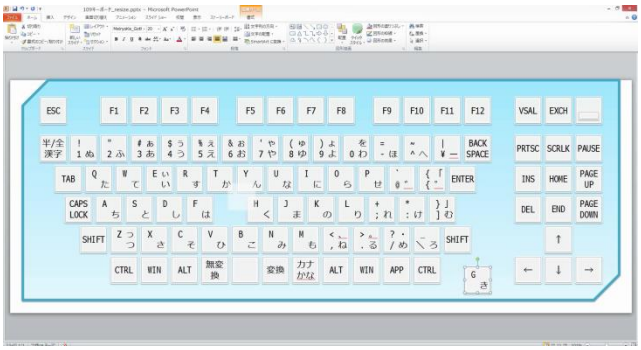

図 5　パワーポイントでのレイアウト編集

3. 評価実験

3.1 実験概要

前節で提案した手法の効果を検証するため，以下の手順で評価実験を行った．

(1) キー入力データの収集

外部ツールを用いて，被験者から「各キーがそれぞれ何回押されたか」のデータを収集する．今回はフリーソフトの「タイプ数カウンター」[5]を使用し，キーのタイプ数を計上した．収集したデータをまとめたものを表 1 に示す．

表 1　収集データ

名称	カウント総数	カウント日数
成人男性 A	616518	54
成人男性 B	16147545	1008

特徴としては，成人男性 A と成人男性 B のデータ量の間に，大きな差があることが挙げられる．

(2) データ形式の変換

上記のデータを Excel に入力し，自作の変換ツールを用いて，先ほどのソフトウェアキーボードが読み込める形式に変換する．

(3) キー配列の最適化

ソフトウェアキーボードを用いて評価を行う．最初に，可視化キーによってキー入力を可視化し，それを画像として記録する．次に，最適化キーによるキーの入れ替えを，評価値 E を記録しながら 50 回行う．本来ならば，1 回入れ替えるごとにそのキー配列に慣れる手順が入るが，今回は時間の都合上省略する．最後に，再び可視化キーによるキー入力の可視化を行い，画像として記録した後，最初のキー配列と比較する．

3.2 実験結果

横軸にキーの入れ替え回数，縦軸に評価値 E を取ったグラフを図 6 に示す．

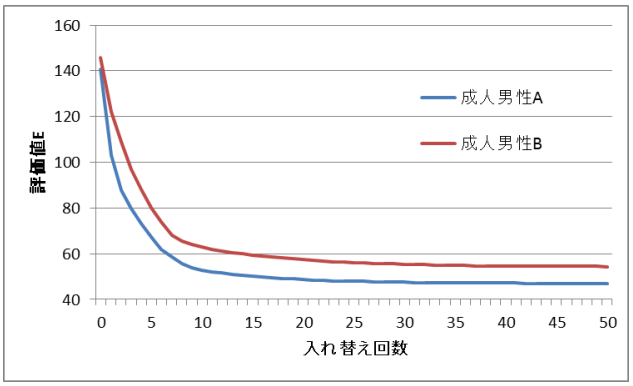

図 6　キーの入れ替え回数と評価値 E

次に，成人男性 A について，キー入力を可視化したものを図 7 と図 8 に示す．さらに，50 回の入れ替えを行った後の，ホームポジション付近のキーの配置を図 9 に示す．

図 7　キー入力の可視化（成人男性 A・入れ替え前）

図 8　キー入力の可視化（成人男性 A・入れ替え後）

図 9　入れ替え後のキー配列（成人男性 A）

同様にして，成人男性 B についても図 10 から図 12 に実験結果を示す．

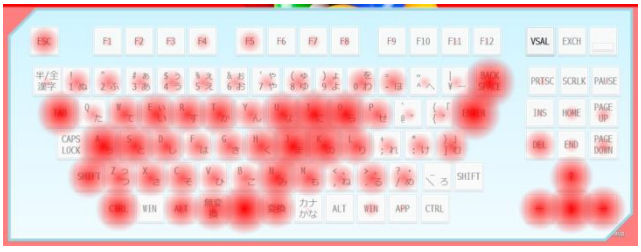

図 10　キー入力の可視化（成人男性 B・入れ替え前）

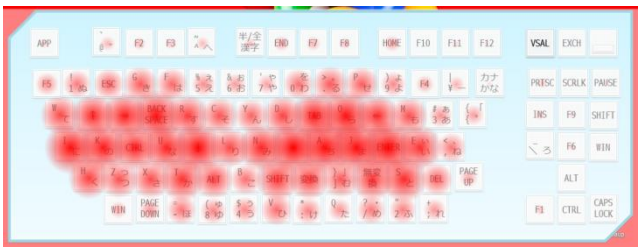

図 11　キー入力の可視化（成人男性 B・入れ替え後）

図 12　入れ替え後のキー配列（成人男性 B）

4.　考察

図 6 より，最初の 5 回程度の入れ替えで，評価値 E が大幅に減少しているのが見て取れる．つまり，キーの配置を最適化するにあたって，大きな効果があるのは最初の数回だけであり，逆に言えば，少ない入れ替え回数で，キー配列はほぼ最適化できるということである．

次に，成人男性 A の図 7 と図 8，成人男性 B の図 10 と図 11 を見てみると，最初はまばらであった濃い赤色の部分が，キーを 50 回入れ替えた後にはホームポジション付近に集まっていることを確認できる．つまり，入力頻度の高いキーがホームポジション付近に集まっているということであり，このことからも適切に最適化されていると言える．

最後に，図 9 と図 12 を比較してみる．細かい配置こそ違うものの，ホームポジション付近に集まるキーには，両者の間で多くの共通点が見て取れる．これの意味するところは，最適なキー配列とは，個人に特有のものではなく，ある程度の共通性を持つ可能性があるということである．

5.　結言

本研究では，経路依存性に起因するキーボードのキー配列の問題を，キー配置の組み合わせ最適化問題と捉え，貪欲法を用いたキーの入れ替えシステムで解決を試みた．結果としては，キーの形状を考慮していないなどの課題もあるものの，少ない入れ替え回数で目的関数を大幅に減少させることが確認できた．

今後の課題として，まずキーの形状を考慮することが挙げられる．今回の評価実験ではすべてのキーを正方形に統一した．その理由は，Enter キーや Shift キーなど，形が特殊なキーをそのまま入れ替えると，他のキーと重なってしまう問題が生じたからである（図 13 参照）．この問題を解決するためにも，キーの大きさを考慮したシステムの構築が必要不可欠である．

図 13　Enter キーと Shift キーが重なる例

次に，指に重みづけを行うことが考えられる．例えば，小指や薬指は，親指や人差し指に比べて動かしづらい．従って，特によく押されるキーは，キーボードの中央付近に配置した方がより低負担で済むと考えられる．また Dvorak 配列では入力時に打鍵が左右交互になるよう工夫がなされている．このように指の性質も考慮してみると，また新たな発見があるように思われる．

また，矢印キーの処遇の検討が挙げられる．矢印キーは使用頻度が多いにも関わらず，ホームポジションから離れた場所にあるため，今回のいずれの被験者でもキー入れ替えの対象となった．しかし，各矢印キーが個々に入れ替えられた結果, 直感的に分かりづらい所に分散してしまった．そのため，矢印キーについては入れ替え対象外とする，4 つまとめて移動する，相対位置を保つ（左矢印キーは左側に置くなど）といった制約を設けることが考えられる．

最後に，データの収集方法が挙げられる．提案手法で最適化を行うためには一定期間キー入力の統計をとる必要があるため，すぐに最適化を行えない．この問題の対策として，キーボードからの入力だけではなく，文章などのテキストコーパスからも出現頻度のデータを収集することが考えられる．

参考文献

[1] 名和小太郎, “QWERTY 配列”, 情報管理, Vol. 47, No. 12, pp. 839-841, 2005.
[2] “経路依存性 - 経済・社会 - 経営用語集 - ビジネス EX”, http://bizex.goo.ne.jp/tool/mng/9_99/5369/ (2016-12-23 参照)
[3] 桜井雄介, 増井俊之, “QWERTY ソフトキーボード上のフリック日本語入力システム”, 情報処理学会研究報告, Vol. 2013-HCI-154, No. 5, pp. 1-4, 2013.
[4] 久野祐輝, 志築文太郎, 田中二郎, “キーを指の設置位置とその周囲に配置するソフトウェアキーボード”, 情報処理学会論文誌, Vol. 55, No. 4, pp.1354-1364, 2014.
[5] “タイプ数カウンター”, http://www.vector.co.jp/soft/win95/util/se399353.html (2016-12-23 参照)