

UMA ABORDAGEM HEURÍSTICA PARA O *SINGLE-FINGER* *KEYBOARD LAYOUT PROBLEM*

Ana Beatriz Herthel

Universidade Federal da Paraíba, Departamento de Engenharia de Produção, Centro de Tecnologia
Campus I, Cidade Universitária, João Pessoa-PB, 58051-970, Brasil.

abherthel@gmail.com

Anand Subramanian

Universidade Federal da Paraíba, Departamento de Sistemas de Computação, Centro de
Informática

Mangabeira, Rua dos Escoteiros s/n, CEP 58059-900, João Pessoa, Brasil.

anand@ci.ufpb.br

RESUMO

O problema associado ao desenvolvimento de *layout* de teclado para um dedo é denominado SK-QAP e definido como variante do *Quadratic Assignment Problem* (QAP), um problema clássico de otimização. Este trabalho aborda o SK-QAP através de um algoritmo *Iterated Local Search* (ILS), denominado ILS-SKQAP, que conta com três estruturas de vizinhança em sua fase de busca local. Duas delas foram anteriormente utilizadas na resolução do SK-QAP e a terceira (*two pairs swap*) adaptada, neste trabalho, de vizinhança do QAP. Além disso, dois mecanismos de perturbação foram desenvolvidos para o ILS-SKQAP: *ejection chain* e *multiple pairwise-exchange*. O ILS-SKQAP foi usado na resolução das 24 instâncias existentes do SK-QAP para os idiomas inglês, francês, italiano e espanhol, obtendo resultados altamente competitivos em termos de qualidade das soluções encontradas e tempos computacionais. Ademais, seis novas instâncias foram desenvolvidas para a língua portuguesa e resolvidas pelo algoritmo.

PALAVRAS CHAVE. *Single-finger. Layout. Teclado. Iterated Local Search.*

Tópicos: MH - Meta-heurísticas

ABSTRACT

The problem associated with the design of a single-finger keyboard layout is denominated SK-QAP and it is defined as a variation of the Quadratic Assignment Problem (QAP), a classical optimization problem. This work approaches the SK-QAP by means of an Iterated Local Search (ILS) algorithm, denominated ILS-SKQAP. Three neighborhood structures were incorporated in the local search phase of the algorithm, in which two of them were previously applied to solve the SK-QAP, whereas two pairs swap was adapted, in this work, from a QAP neighborhood. Furthermore, two perturbation mechanisms were developed for the ILS-SKQAP: *ejection chain* and *multiple pairwise-exchange*. ILS-SKQAP was used to solve the 24 existing instances for English, French, Italian and Spanish languages with highly competitive results both in terms of solution quality and CPU time. Moreover, a set of six instances for the Portuguese language was developed and solved by the algorithm.

KEYWORDS. *Single-finger. Layout. Keyboard. Iterated Local Search.*

Paper topics: MH - Metaheuristics

1. Introdução

O *layout* QWERTY foi desenvolvido com o intuito de causar menos obstrução nos mecanismos de máquinas de escrever, ao promover diminuição da velocidade de digitação. Para isso, os pares de caracteres frequentemente escritos em sequência foram posicionados em lados opostos do teclado, facilitando o uso das duas mãos [Alswaidan et al., 2015; Bi et al., 2012; MacKenzie et al., 1999; Yamada, 1980]. Quando se considera o uso de teclados virtuais em *smartphones*, *tablets* e terminais de autoatendimento, é comum o uso de um dedo para digitação e, digitar desta forma em um *layout* destinado à alternância de mãos, faz com que o dedo passe constantemente de um lado para outro do teclado, em quantidade excessiva de movimentos. Dessa forma, um *layout* para teclados virtuais, destinados ao uso com um dedo, deve ser desenvolvido tendendo-se a aproximar os caracteres cujos pares ocorrem com mais frequência, diminuindo o tempo de movimentação de uma tecla para outra ao digitar um texto.

O problema de otimização que busca definir a disposição dos caracteres nas teclas de um teclado para um dedo é conhecido como *single-finger keyboard layout problem* (SK-QAP) e foi proposto por [Dell'Amico et al., 2009] como uma variante do *Quadratic Assignment Problem* (QAP), já que permite a existência de locais vazios. Os problemas QAP e SK-QAP são considerados NP-difícil [Sahni e Gonzalez, 1976]; [Dell'Amico et al., 2009] e, para resolver o SK-QAP neste trabalho, decidiu-se pela implementação de uma meta-heurística baseada em busca local, uma vez que se trata de uma abordagem simples e robusta capaz de obter soluções de alta qualidade em tempos computacionais razoáveis [Focacci et al., 2004; Lourenço et al., 2010; Osman e Laporte, 1996].

No trabalho de [Dell'Amico et al., 2009] foram desenvolvidos teclados otimizados para os idiomas inglês, francês, italiano e espanhol. De acordo com dados do [Ministério da Cultura, 2017], o português é a quarta língua mais falada do mundo, com 260 milhões de falantes e, por isso há a necessidade de um teclado otimizado para este idioma, uma lacuna nas pesquisas relacionadas ao desenvolvimento *layouts* para teclados para um dedo. Sendo assim, este trabalho pretende também desenvolver um conjunto de instâncias para a língua portuguesa de forma a suprir essa necessidade.

A Tabela 1 lista trabalhos que abordam o problema de alocação de caracteres a teclas, especificando seus autores, ano de publicação, tipo de teclado abordado na pesquisa e método utilizado para resolver o problema. Os tipos de teclado abordados são: destinados a um dedo (*s-finger*), a vários dedos (*n-finger*), a dois dedos (polegares) (*2-finger*), com um caractere por tecla (*s-char*) e com múltiplos caracteres por tecla (*m-char*).

Tabela 1: Trabalhos relacionados que abordam *layouts* de teclados

Autores	Ano	Teclado abordado	Método
Mackenzie e Zhang	1999	<i>s-finger</i>	Método Iterativo
Zhai, Hunter e Smith	2002	<i>s-finger</i>	Simulação; Algoritmo Metropolis
Raynal e Vigouroux	2005	<i>s-finger</i>	GA
Li, Chen e Goonetilleke	2006	<i>s-finger</i>	Algoritmo Guloso + SA
Dell'Amico et al.	2009	<i>s-finger</i>	LS; SA; TS; VNS; FANT
Bi, Smith e Zhai	2010	<i>s-finger</i>	Algoritmo Metropolis
Bi, Smith e Zhai	2012	<i>s-finger</i>	Algoritmo Metropolis
Dunlop e Levine	2012	<i>s-finger</i>	Fronteira de Pareto
Alswaidan, Hosny e Najjar	2014	<i>s-finger</i>	GA
Murali e Panicker	2016	<i>s-finger</i>	GA + TOPSIS
Eggers et al.	2003	<i>n-finger</i>	ACO
Yin e Su	2011	<i>s-char; m-char;</i> <i>s-finger; n-finger</i>	Cyber Swarm
Oulasvirta et al.	2013	<i>2-finger</i>	Gradient descent + SA
Karrenbauer e Oulasvirta	2014	<i>s-finger; n-finger</i>	Branch-and-Bound
Işeri e Ekşioğlu	2015	<i>n-finger</i>	GA híbrido

Este trabalho tem como objetivo desenvolver uma meta-heurística baseada em busca local

para resolver o SK-QAP, assim como gerar e resolver instâncias do SK-QAP para a língua portuguesa, produzindo, para este idioma, um teclado otimizado para um dedo.

O restante deste trabalho será estruturado como se segue: A Seção 2 apresenta a formulação matemática e definição do SK-QAP. Na Seção 3 um algoritmo baseado em busca local, para resolução do SK-QAP, é proposto, apresentando-se as estruturas de vizinhança e perturbações incorporadas ao procedimento. O processo de geração de instâncias para a língua portuguesa também é apresentado nesta seção. A Seção 4 contém os resultados computacionais e a Seção 5 relata as conclusões do trabalho e sugestões para trabalhos futuros.

2. Descrição do Problema

No SK-QAP, há um conjunto $C = 1, 2, \dots, n$ de caracteres a serem alocados a um conjunto $M = 1, 2, \dots, e = p \times q$ de teclas (locais), com p e q suficientemente grandes para comportar os n caracteres em sua totalidade, de maneira que cada caractere seja designado a uma tecla e cada tecla receba, no máximo, um caractere. Todas as teclas são iguais e de mesmo tamanho, de formato retangular, distribuídas em uma rede de dimensão e , contendo caracteres diferentes em todas elas. Existe, também, uma matriz F de fluxos f_{ij} entre pares de caracteres i e j e uma matriz D contendo valor da função de Fitts d_{kl} para as teclas k e l . As variáveis de decisão são x_{ik} e x_{jl} que assumem valor 1 quando o caractere $i(j)$ é alocado na tecla $k(l)$, e 0, caso contrário.

A lei de Fitts [Fitts, 1954] computa tempo e dificuldades durante a realização de um movimento para uma determinada área, de maneira que o tempo necessário para mover-se de um ponto inicial a um ponto final é proporcional a um índice de dificuldade. A formulação mostrada na Equação (1) calcula o tempo de movimentação TM e é mais comumente utilizada nas pesquisas relacionadas ao SK-QAP.

$$TM = a + b * \log_2 \left(\frac{S}{A} + 1 \right) \quad (1)$$

Nesta formulação, a e b são constantes de valores pré-fixados, S é a distância entre os centros de duas teclas e A , a área de cada tecla [Alswaidan et al., 2015; Dell’Amico et al., 2009; Fitts, 1954; Li et al., 2006; MacKenzie et al., 1991; MacKenzie e Zhang, 1999; Zhai et al., 2002]. A lei de Fitts (Equação (1)) não leva em consideração o tempo de reação de uma pessoa ao digitar, considerando-se, portanto, que o digitador é experiente e tem conhecimento do *layout* do teclado utilizado.

O SK-QAP pode ser formulado, então, da seguinte maneira:

$$z = \min \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^e \sum_{l=1}^e f_{ij} d_{kl} x_{ik} x_{jl} \quad (2)$$

Sujeito a:

$$\sum_{k=1}^e x_{ik} = 1 \quad i \in C \quad (3)$$

$$\sum_{i=1}^n x_{ik} \leq 1 \quad k \in M \quad (4)$$

$$x_{ik} \in \{0, 1\} \quad i \in C, k \in M. \quad (5)$$

A função objetivo (2) minimiza o produto do fluxo entre os caracteres i e j pelo valor da função de Fitts (Equação (1)) entre as teclas k e l pelas variáveis de decisão x_{ik} e x_{jl} . As restrições (3) definem que cada caractere precisa, necessariamente, estar alocado a uma tecla. As restrições (4) especificam que nem toda tecla precisa ter um caractere associado a ela, porém um caractere é o máximo que pode ser alocado nesse espaço. Já o conjunto de restrições (5) define os possíveis valores assumidos pelas variáveis de decisão.

3. Algoritmo Proposto

O algoritmo proposto é baseado no método ILS e denominado ILS-SKQAP, cujo funcionamento está descrito no Algoritmo 1. Os parâmetros recebidos pelo algoritmos são o número máximo de *restarts* (I_{MS}) e o número máximo de iterações do procedimento ILS (I_{ILS}). O ILS-SKQAP tem como primeiros passos a inicialização da função objetivo e da solução (linhas 2 e 3). Em seguida, inicia-se o laço de *restarts* (linha 4), em que, a cada iteração $Iter_{MS}$, uma solução inicial é gerada (linha 5) e atribuída à solução atual (linha 6) e a iteração atual do procedimento ILS ($Iter_{ILS}$) recebe valor 0 (linha 7). Os próximos passos (linhas 8-15) denotam a fase ILS do algoritmo, na qual o método de busca local é aplicado à solução atual (linha 9) e, se houver melhora no valor da função objetivo, seu novo valor e a solução melhorada são armazenados e $Iter_{ILS}$ recebe o valor 0 (linhas 10-13). O ILS continua com a aplicação de perturbação à melhor solução encontrada no *restart* corrente e incremento de 1 a $Iter_{ILS}$ (linhas 14 e 15). Se, ao final do ILS houver melhora no valor da função objetivo, este é armazenado, bem como a solução associada a ele (linhas 16-18). A melhor solução é, então, retornada pelo algoritmo (linha 19).

Algoritmo 1 ILS-SKQAP

```
1: Procedimento ILS-SKQAP( $I_{MS}, I_{ILS}$ )
2:    $f^* \leftarrow \infty$ 
3:    $s^* \leftarrow \emptyset$ 
4:   para  $Iter_{MS} = 1, \dots, I_{MS}$  faça
5:      $s \leftarrow \text{GeraTecladoInicial}()$ 
6:      $s' \leftarrow s$ 
7:      $Iter_{ILS} \leftarrow 0$ 
8:     enquanto  $Iter_{ILS} \leq I_{ILS}$  faça
9:        $s \leftarrow \text{BuscaLocal}(s)$ 
10:      se  $f(s) < f(s')$  então
11:         $s' \leftarrow s$ 
12:         $f(s') \leftarrow f(s)$ 
13:         $Iter_{ILS} \leftarrow 0$ 
14:       $s \leftarrow \text{Perturbação}(s')$ 
15:       $Iter_{ILS} \leftarrow Iter_{ILS} + 1$ 
16:      se  $f(s') < f^*$  então
17:         $s^* \leftarrow s'$ 
18:         $f^* \leftarrow f(s')$ 
19:      retorne  $s^*$ 
20: fim Procedimento
```

3.1. Geração da Solução Inicial

O procedimento construtivo desenvolvido nesse trabalho possui componentes aleatórios e gulosos. A aleatoriedade se faz necessária para assegurar mais detalhada exploração do espaço de busca ao promover diferentes estados iniciais e o componente guloso garante menor esforço nas fases iniciais da busca local posteriormente aplicada.

O Algoritmo 2 descreve o funcionamento do procedimento de construção da solução inicial. Os primeiros passos do procedimento são a inicialização da lista L de caracteres candidatos à alocação (linha 2), da matriz de teclas M , que deve estar vazia, a princípio (linha 3), e da solução s (linha 4). Em seguida, determina-se a tecla central da matriz M (linhas 5 e 6) que recebe o caractere espaço o qual é, então, retirado de L (linhas 7 e 8). Esses passos foram incorporados ao algoritmo pela observação anterior de que o caractere espaço é o mais frequente em todas as instâncias, o que o torna um candidato propício para o centro do teclado, localizando-se a menores distâncias dos outros caracteres. Os próximos passos do algoritmo introduzem o componente aleatório do procedimento. Quatro caracteres (c_1 a c_4) são escolhidos, aleatoriamente, em L e alocados nas teclas imediatamente acima, abaixo, à direita e à esquerda da tecla central, sendo posteriormente excluídos da lista de candidatos (linhas 9 a 14). O custo preliminar da solução com essas 5 alocações é

então calculado na linha 15. As linhas 16 a 26 descrevem o componente guloso do algoritmo, no qual os caracteres restantes de L vão sendo escolhidos, um a um, e testados nas teclas vazias de M , sendo guardado o melhor custo obtido dentre os testes de alocação e a tecla associada a este custo. Após todas as teclas terem sido testadas, o caractere cujo teste retornou o menor custo é alocado na posição correspondente e retirado de L . Esses passos se repetem até que todos os caracteres tenham sido alocados e L esteja vazia. O procedimento termina com a atribuição de M à solução s (linha 27) que é retornada pelo algoritmo (linha 28).

Algoritmo 2 Gerar Teclado Inicial

```

1: Procedimento GERATECLADOINICIAL
2:   Inicialize  $L$ ;                                ▷ Lista de caracteres candidatos
3:   Inicialize  $M(p \times q)$  com  $-1$ ;                ▷ Matriz de teclas
4:    $s \leftarrow \emptyset$ ;
5:    $L_c \leftarrow (p/2)$ ;                            ▷ Linha central
6:    $C_c \leftarrow (q/2)$ ;                            ▷ Coluna central
7:    $M(L_c, C_c) \leftarrow \text{espaco}$ ;
8:    $L \leftarrow L - \{\text{espaco}\}$ ;
9:   Selecione 4 caracteres aleatoriamente  $(c_1, c_2, c_3, c_4)$ ;
10:   $M(L_c, (C_c - 1)) \leftarrow c_1$ ;
11:   $M(L_c, (C_c + 1)) \leftarrow c_2$ ;
12:   $M((L_c - 1), C_c) \leftarrow c_3$ ;
13:   $M((L_c + 1), C_c) \leftarrow c_4$ ;
14:   $L \leftarrow L - \{c_1, c_2, c_3, c_4\}$ ;
15:  Calcule  $iniCost$ ;                                ▷ Custo do estado inicial da solução
16:  enquanto  $L$  não está vazia faça
17:     $MinCost \leftarrow \infty$ ;
18:    para  $i = 0, \dots, p$  faça
19:      para  $j = 0, \dots, q$  faça
20:        para  $c = 0, \dots, |L| - 1$  faça
21:           $CurrCost \leftarrow iniCost + \text{custo de inserção de } c \text{ em } M(i, j)$ ;
22:          se  $CurrCost < MinCost$  então
23:            Guarde os valores de  $c, i$  e  $j$ ;
24:             $MinCost \leftarrow CurrCost$ ;
25:           $M(i, j) \leftarrow c$ ;
26:           $L \leftarrow L - \{c\}$ ;
27:   $s \leftarrow M$ ;
28:  retorne  $s$ ;
29: fim Procedimento

```

3.2. Busca Local

O algoritmo de busca local proposto utiliza um esquema baseado no procedimento *variable neighborhood descent* (VND) proposto por [Mladenović e Hansen, 1997], com ordem aleatória na escolha de vizinhanças. Esse algoritmo funciona através de uma lista de estruturas de vizinhança, em que uma delas é escolhida, aleatoriamente. Se a nova solução obtida após ação dessa vizinhança trouxer melhoras ao valor da função objetivo, todas as estruturas da lista ficam disponíveis na próxima iteração. Caso contrário, a estrutura explorada é retirada da lista de vizinhanças acessíveis na iteração seguinte e nova estrutura é aleatoriamente escolhida. O procedimento termina se todas as estruturas de vizinhança foram exploradas mas o valor da função objetivo não sofreu melhoras.

Nesse estudo, três estruturas de vizinhança foram utilizadas. Duas delas foram incorporadas a partir do trabalho de [Dell'Amico et al., 2009]: *contour filling* ($N^{(1)}$) e *pairwise-exchange* ($N^{(2)}$). A vizinhança $N^{(1)}$ funciona através do movimento de um caractere para uma posição de contorno. Nesse caso, uma posição de contorno é definida como uma tecla vazia em cuja adjacência há pelo menos uma tecla com caractere alocado. A vizinhança $N^{(2)}$ seleciona dois caracteres e realiza a troca de suas posições no teclado.

Para computar a variação do valor da função objetivo de maneira eficiente em $N^{(1)}$, foram usadas as Equações (6) e (7), como apresentadas por [Dell'Amico et al., 2009]. A Equação (6) define o cálculo da variação de valor da função objetivo ao mover-se um caractere i para uma posição de contorno j , gerando uma solução φ' a partir de uma solução φ . Já a Equação (7), calcula a variação do valor da função objetivo após movimentação de um caractere h para uma posição de contorno l , gerando a solução φ'' a partir de φ' . A condição para o uso da Equação (7) é que a posição de contorno l pertença ao conjunto de posições de contorno de φ e φ' . Caso se trate de uma posição de contorno que surge após a movimentação de i para j , a Equação (6) deve ser usada. Através da Equação (6), $z(\varphi')$ é calculada em $O(n)$ operações e, com a Equação (7), $z(\varphi'')$ é computada com complexidade $O(1)$ [Dell'Amico et al., 2009]. Para tirar vantagem dessas equações e reduzir a complexidade de cálculo da função objetivo em um movimento seguinte de $N^{(1)}$, esta vizinhança foi implementada com o método de primeiro vizinho com melhora, em que os movimentos são realizados com base na primeira melhora obtida.

$$\Delta(\varphi, i, j) = z(\varphi') - z(\varphi) = \sum_{\substack{k=1 \\ k \neq i}}^n f_{ki}(d_{\varphi_k j} - d_{\varphi_k \varphi_i}) \quad (6)$$

$$\Delta(\varphi', h, l) = z(\varphi'') - z(\varphi') = \Delta(\varphi, h, l) + f_{ih}(d_{jl} - d_{\varphi_i l} + d_{\varphi_i \varphi_h} - d_{\varphi_h j}) \quad (7)$$

A variação de valor da função objetivo em $N^{(2)}$ foi eficientemente calculada usando a Equação (8), também apresentada no trabalho de [Dell'Amico et al., 2009]. Nesse caso, φ' é uma solução obtida a partir de φ após troca de posições entre um par de caracteres r e s . Com o uso da Equação (8), $z(\varphi')$ é computado em $O(n)$ operações.

$$\Delta(\varphi, r, s) = z(\varphi') - z(\varphi) = \sum_{\substack{k=1 \\ k \neq r, s}}^n (f_{rk} - f_{sk})(d_{\varphi_s \varphi_k} - d_{\varphi_r \varphi_k}) \quad (8)$$

A terceira estrutura de vizinhança utilizada foi desenvolvida neste trabalho e denominada *two pairs swap* ($N^{(3)}$). Seu funcionamento se dá por meio da seleção de dois pares de caracteres adjacentes e troca de suas posições no teclado. Os pares de caracteres podem ser adjacentes no sentido vertical ou horizontal e suas posições relativas são mantidas após a realização da troca. O cálculo eficiente da variação do valor da função objetivo em $N^{(3)}$ foi realizado através da adaptação de equação proposta por [Frieze et al., 1989] para o QAP. No caso do SK-QAP, φ' é a solução obtida através da troca entre caracteres r e s e o valor de $z(\varphi')$ é calculado usando a Equação (8). A troca entre caracteres u e v gera uma solução φ'' e a Equação (9) é utilizada para calcular $z(\varphi'')$ com complexidade $O(1)$.

$$\begin{aligned} \Delta(\varphi', u, v) &= z(\varphi'') - z(\varphi') \\ &= \Delta(\varphi, u, v) + (f_{ru} - f_{rv} + f_{sv} - f_{su})(d_{\varphi_s \varphi_u} - d_{\varphi_s \varphi_v} + d_{\varphi_r \varphi_v} - d_{\varphi_r \varphi_u}) \end{aligned} \quad (9)$$

As estruturas de vizinhança $N^{(2)}$ e $N^{(3)}$ utilizam o método do melhor vizinho, no qual se realiza apenas o movimento que gera a melhora mais vantajosa à função objetivo. Nas três vizinhanças, o procedimento de atualização da estrutura da solução é feito em tempo constante.

3.3. Mecanismos de Perturbação

Os mecanismos de perturbação desenvolvidos neste trabalho foram denominados *ejection chain* ($P^{(1)}$) e *multiple pairwise-exchange* ($P^{(2)}$). A perturbação $P^{(1)}$ seleciona, aleatoriamente, um conjunto de n caracteres e troca suas posições entre eles, dessa forma: o primeiro caractere selecionado assume a tecla do próximo e assim sucessivamente, até o último caractere assumir a posição anteriormente ocupada pelo primeiro. A perturbação $P^{(2)}$ seleciona n pares de caracteres e troca suas posições. Para $P^{(1)}$ limitou-se o valor de n de 3 a 5 caracteres. Já para $P^{(2)}$, os valores limitantes de n foram de 2 a 3 pares.

3.4. Geração de Instâncias da Língua Portuguesa

Os conjuntos de instâncias para inglês, espanhol, italiano e francês do SK-QAP foram originalmente desenvolvidas no trabalho de [Dell'Amico et al., 2009] e, seu procedimento de geração, adaptado neste trabalho para obter as instâncias da língua portuguesa.

A instância real do idioma português foi gerada a partir da listagem das 10000 palavras mais frequentes da língua, de acordo com o corpus brasileiro, disponibilizado pela PUC-SP (<http://corpusbrasileiro.pucsp.br/>). Partindo dessa lista, a frequência de cada par de caracteres consecutivos foi computada, considerando que a ordem de caracteres em um par não é diferenciadora, ou seja, a frequência para um par “AE”, por exemplo, é somada àquela de um par “EA”. Para computar o caractere espaço, foi considerada a presença do mesmo antes e depois de cada palavra da lista. A instância real do idioma português foi denominada Portuguese.

Um conjunto de 5 instâncias aleatórias também foi criado ao listar as 5000 palavras mais frequentes de acordo com o corpus brasileiro, selecionando-se aleatoriamente, 1000 palavras e suas frequências nesta lista. A partir da nova relação de palavras, realizou-se processo similar à geração da instância real, computando os pares de caracteres encontrados e suas frequências. As instâncias aleatórias da língua portuguesa foram denominadas Portuguese_Rand_1 a 5.

Os caracteres acentuados presentes na lista de palavras foram computados como caracteres diferentes daqueles sem o diacrítico. A decisão foi tomada com base no formato encontrado nas instâncias existentes do problema.

4. Resultados Computacionais

O ILS-SKQAP foi implementado em C++ (g++ 5.4.0) e os experimentos executados em um Intel®Core™ i7-3770 com 3,40 GHz e 16 GB de memória RAM, em sistema operacional Ubuntu Linux 16.04.

4.1. Impacto da Vizinhança *Two Pairs Swap*

Para analisar o impacto da vizinhança *two pairs swap* foi realizado um teste com duas etapas. Durante a primeira etapa foram utilizadas as vizinhanças $N^{(1)}$ e $N^{(2)}$, ao passo que, na segunda, a vizinhança $N^{(3)}$ foi adicionada às configurações anteriores.

Para esse teste, os valores dos parâmetros I_{MS} e I_{ILS} foram definidos como 1 e a fase de perturbações do algoritmo foi ignorada. Observou-se que, ao adicionar a estrutura de vizinhança $N^{(3)}$, há melhor desempenho do algoritmo em 75% das instâncias, ainda com tempos computacionais muito próximos àqueles obtidos nos testes sem esta estrutura, registrando até tempos menores em 33,3% das instâncias. Ao considerar-se classes de instâncias, agrupando-as por idiomas, é possível perceber que ocorrem melhoras nos *gaps* obtidos em todos os casos. A Figura 1 ilustra os *gaps* médios obtidos para cada classe de instâncias considerando as duas etapas do teste.

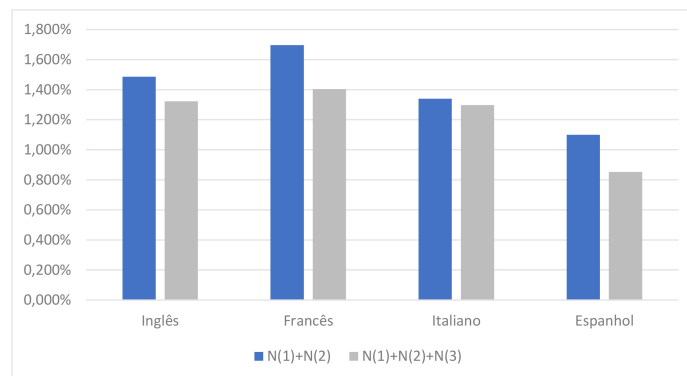


Figura 1: *Gaps* médios das soluções no teste de impacto de $N^{(3)}$

Através de análise dos resultados, optou-se por incorporar a estrutura de vizinhança ao ILS-SKQAP, já que seu comportamento na obtenção de *gaps* foi bastante promissor, sem comprometimento significativo à velocidade do algoritmo.

4.2. Impacto dos Mecanismos de Perturbação

Um teste composto por três etapas foi conduzido para determinar o impacto dos mecanismos de perturbação desenvolvidos para o algoritmo. A primeira etapa utilizou somente a perturbação *ejection chain* ($P^{(1)}$), a segunda etapa usou apenas a *multiple pairwise-exchange* ($P^{(2)}$) e, finalmente, a terceira etapa do teste empregou ambos os mecanismos ($P^{(1)} + P^{(2)}$). Dessa maneira, foi possível avaliar o funcionamento do algoritmo nos casos dos mecanismos de perturbação individuais, bem como no caso em que há a combinação de ambos.

Para este teste o parâmetro I_{ILS} recebeu valor igual ao tamanho de cada instância (n) e I_{MS} recebeu valor 1. Percebeu-se que, em 50% das instâncias do SK-QAP, a utilização de ambos os mecanismos de perturbação oferece menores *gaps* médios entre as soluções obtidas e as melhores soluções reportadas na literatura. As etapas contendo apenas as perturbações individuais resultaram em menores *gaps* para 25% das instâncias, em cada caso. A Figura 2 ilustra os *gaps* médios obtidos para cada classe de instâncias considerando as três etapas do teste.

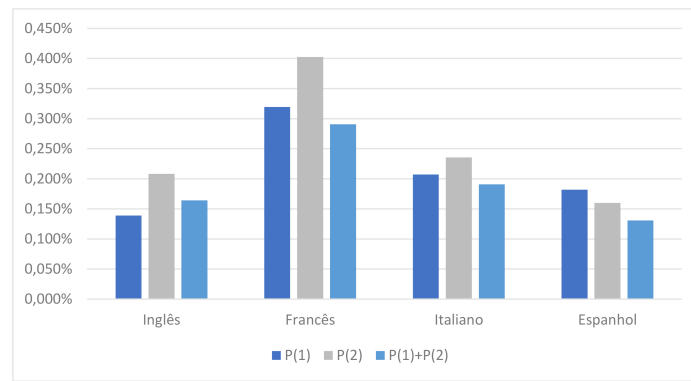


Figura 2: *Gaps* médios no teste de impacto de $P^{(1)}$ e $P^{(2)}$

Mediante análise destes resultados, optou-se por incorporar ambos mecanismos de perturbação propostos ao ILS-SKQAP, visto que foi registrado menor *gap* médio com $P^{(1)} + P^{(2)}$ em três dos quatro conjuntos de instâncias disponíveis, em tempos computacionais competitivos.

4.3. Calibração de Parâmetros

Para calibrar os parâmetros do ILS-SKQAP, foram conduzidos 6 testes com diferentes combinações de I_{MS} e I_{ILS} , a fim de estudar o comportamento do algoritmo em termos de *gap* médio das soluções e tempo computacional demandado. Para cada configuração, o algoritmo foi executado 10 vezes. O procedimento de calibração de parâmetros utilizou apenas o conjunto de instâncias em francês, pois se trata do conjunto de maior tamanho e maior dificuldade de resolução.

As Figuras 3 e 4 apresentam os *gaps* encontrados entre os custos das soluções e o tempo computacional médio para cada combinação de parâmetros testada, respectivamente.

Através destes resultados, definiu-se os valores de I_{MS} e I_{ILS} como 15 e $4n$, respectivamente. Essa combinação oferece menores *gaps*, sempre abaixo de 0,002% e proporciona tempos computacionais competitivos, com um pior caso de 19,36 segundos para a instância French.

4.4. Resultados para as Instâncias Existentes

As instâncias existentes do SK-QAP foram propostas no trabalho de [Dell'Amico et al., 2009] e resolvidas pelos autores através de algoritmos SA, TS, VNS e FANT. Os melhores resultados foram obtidos com o FANT, com parâmetro de reforço da trilha de feromônios R igual a 8, sendo capaz de encontrar as melhores soluções das 24 instâncias em um tempo de execução máximo de 120 segundos.

A Tabela 2 reúne os resultados obtidos pelo ILS-SKQAP, com a parametrização selecionada de $I_{MS} = 15$ e $I_{ILS} = 4n$, para as instâncias existentes dos idiomas inglês, francês, italiano e espanhol. A Tabela 2 tem 7 colunas, das quais, **Instance** lista as 24 instâncias do SK-QAP; **BKS**

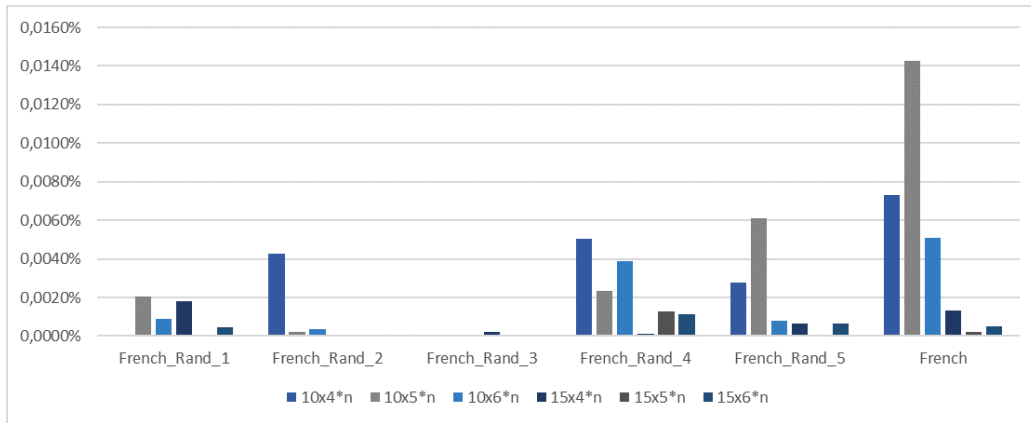


Figura 3: *Gaps* médios das soluções nos testes de calibração de parâmetros

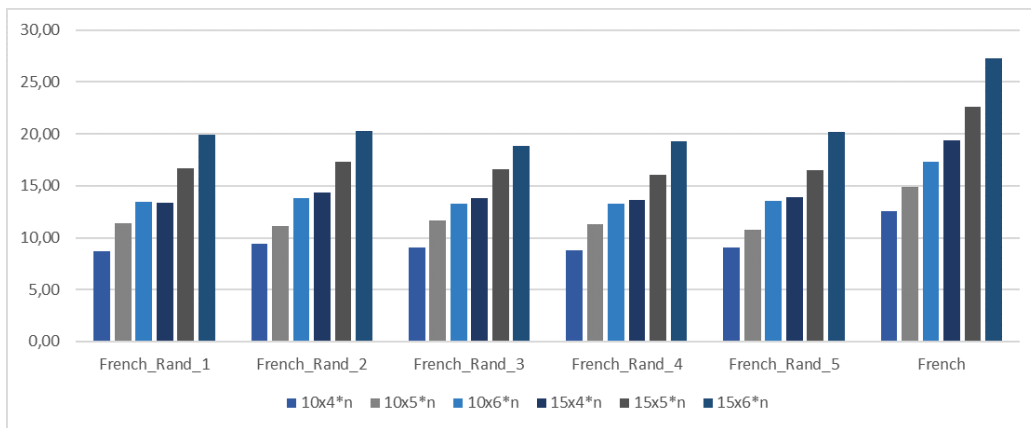


Figura 4: Tempos médios de CPU (s) nos testes de calibração de parâmetros

relaciona as melhores soluções conhecidas para cada instância; **ILS-SKQAP** contém os valores associados às melhores soluções encontradas pelo ILS-SKQAP; **Avg. Sol.**, **Avg. Time (s)** e **Avg. Gap (%)** listam as médias dos valores associados às melhores soluções encontradas, dos tempos computacionais demandados, em segundos e dos *gaps* obtidos entre os valores associados às melhores soluções encontradas pelo ILS-SKQAP e aqueles associados às melhores soluções conhecidas, respectivamente, considerando 10 execuções do algoritmo. Por fim, **#Best** traz o número de vezes em que o ILS-SKQAP encontrou a melhor solução conhecida, em 10 execuções.

O *site* CPU Benchmark (<https://www.cpubenchmark.net/>) foi usado para comparar melhor a eficiência do ILS-SKQAP com outros resultados para as mesmas instâncias, através da estimação de fator comparativo de CPU com base em valores de desempenho por uma única *thread*. Estimou-se que o Intel Pentium 4 3,0 GHz utilizado no trabalho de [Dell'Amico et al., 2009] é 3,18 vezes mais lento que o Intel i7 3,40 GHz utilizado neste trabalho. Através desse fator, o tempo máximo de execução determinado por [Dell'Amico et al., 2009] seria traduzido em de 37,73 segundos, que é um valor 94,9% maior que o pior tempo de 19,36 segundos alcançado pelas configurações adotadas para o ILS-SKQAP. Pode-se perceber, que o ILS-SKQAP tem desempenho superior, em termos de custo de CPU, alcançando média de 14,75 segundos para as instâncias do idioma francês, a classe mais difícil. O algoritmo proposto demonstrou robustez ao ser capaz de alcançar as melhores soluções conhecidas de todas as 24 instâncias, na maioria das execuções, obtendo *gaps* médios menores que 0,001% em todas as classes de instâncias, com destaque para *gaps* médios inferiores a 0,0007% quando agrupadas as instâncias do idioma francês.

Tabela 2: Resultados do ILS-SKQAP por instância existente

Instance	BKS	ILS-SKQAP	Avg. Sol.	Avg. Time (s)	Avg. Gap (%)	#Best
English	1199070166	1199070166	1199070166	4,074	0,000	10
English_Rand_1	647405083	647405083	647405083	2,413	0,000	10
English_Rand_2	655664577	655664577	655666606	3,104	<0,001	7
English_Rand_3	652514886	652514886	652514886	2,979	0,000	10
English_Rand_4	656764071	656764071	656764071	2,699	0,000	10
English_Rand_5	654557377	654557377	654557377	2,871	0,000	10
French	13491323058	13491323058	13491501777	19,364	0,001	6
French_Rand_1	621603962	621603962	621615197	13,378	0,002	7
French_Rand_2	624613625	624613625	624613657	14,383	<0,001	9
French_Rand_3	622277343	622277343	622278607	13,796	<0,001	8
French_Rand_4	622715248	622715248	622715905	13,650	<0,001	9
French_Rand_5	621554539	621554539	621558662	13,935	<0,001	8
Italian	4025754990	4025754990	4025754990	8,746	0,000	10
Italian_Rand_1	2626963740	2626963740	2626963740	7,459	0,000	10
Italian_Rand_2	2632639438	2632639438	2632639438	8,066	0,000	10
Italian_Rand_3	2623137727	2623137727	2623137727	7,954	0,000	10
Italian_Rand_4	2629747342	2629747342	2629747342	8,252	0,000	10
Italian_Rand_5	2626060706	2626060706	2626060706	8,294	0,000	10
Spanish	1229359070	1229359070	1229359070	8,364	0,000	10
Spanish_Rand_1	823360277	823360277	823360277	5,560	0,000	10
Spanish_Rand_2	826559542	826559542	826559542	5,911	0,000	10
Spanish_Rand_3	822666809	822666809	822666809	5,984	0,000	10
Spanish_Rand_4	828720885	828720885	828720885	6,003	0,000	10
Spanish_Rand_5	824672158	824672158	824672158	5,503	0,000	10

4.5. Resultados para as Instâncias da Língua Portuguesa

A Tabela 3 reúne os resultados obtidos pelo ILS-SKQAP, com a parametrização selecionada de $I_{MS} = 15$ e $I_{ILS} = 4n$, para as instâncias da língua portuguesa desenvolvidas neste trabalho.

Tabela 3: Resultados do ILS-SKQAP por instância da língua portuguesa

Instance	ILS-SKQAP	Avg. Sol.	Avg. Time (s)	Avg. Gap (%)	#Best
Portuguese	1123847642543	1123847642543	16,812	0,000	10
Portuguese_Rand_1	157338467337	157338467337	16,555	0,000	10
Portuguese_Rand_2	153688782196	153695204883	16,592	0,004	6
Portuguese_Rand_3	185760833068	185764279729	17,762	0,002	6
Portuguese_Rand_4	155889624542	155889624542	16,514	0,000	10
Portuguese_Rand_5	162264039418	162268123369	16,457	0,003	3

Pode-se perceber que o ILS-SKQAP também proporcionou bons resultados para as instâncias da língua portuguesa, alcançando a melhor solução em todas elas e obtendo *gaps* médios inferiores a 0,005%, com tempos computacionais de 16,78 segundos, em média, para esta classe. O melhor *layout* obtido para a instância Portuguese está ilustrado na Figura 5.

5. Considerações Finais

Este trabalho aborda o *single-finger keyboard layout problem* (SK-QAP), originalmente proposto por [Dell'Amico et al., 2009]. Estabeleceu-se o uso da meta-heurística ILS como método para resolver o SK-QAP devido à sua simplicidade de atuação. O algoritmo proposto foi denominado ILS-SKQAP e inicia-se com um procedimento construtivo constituído de componentes gulosos e aleatórios. Na fase de busca local, o ILS-SKQAP usa um procedimento baseado em VND, com ordem aleatória na seleção de vizinhanças, composto por três estruturas de vizinhança e, para escapar de ótimos locais, duas estratégias de perturbação são incorporadas.

		K	J	X	Ó		
	Ú	Q	U	L	G	Ê	
Y	B	M	E	R	T	Á	
-	É	S	Ø	A	N	H	Ô
	À	P	O	D	I	Z	
	W	F	Ã	C	V	Â	
		Õ	Ç	Í			

Figura 5: Teclado em português

O ILS-SKQAP foi testado nas 24 instâncias propostas por [Dell’Amico et al., 2009] para os idiomas inglês, francês, italiano e espanhol, mostrando-se eficiente na resolução do SK-QAP ao proporcionar resultados extremamente competitivos, registrando *gaps* médios inferiores a 0,001% por classe de instância, com tempos computacionais, em média, inferiores a 20 segundos. Na maioria das execuções do algoritmo, a melhor solução reportada das instâncias existentes foi encontrada.

Um conjunto de 6 novas instâncias foi apresentado para o SK-QAP, adicionando o português à lista de idiomas abordados para este problema. O algoritmo ILS-SKQAP foi capaz de resolvê-las, gerando um *layout* para usuários de teclados de língua portuguesa, a partir da instância real Portuguese.

Para trabalhos futuros, sugere-se desenvolver experimentos práticos com usuários, de maneira a avaliar suas experiências em aspectos como tempo e esforço para aprender um novo *layout*, velocidade e taxa de erros de digitação e conforto ao digitar. Novas instâncias para outras línguas também devem ser desenvolvidas para ampliar a variedade de teclados otimizados disponíveis ao público.

Referências

- Alswaidan, N., Hosny, M. I., e Najjar, A. B. (2015). A genetic algorithm approach for optimizing a single-finger arabic keyboard layout. In Arai, K., Kapoor, S., e Bhatia, R., editors, *Intelligent Systems in Science and Information 2014*, p. 261–277, Cham. Springer International Publishing. ISBN 978-3-319-14654-6. URL <https://doi.org/10.1109/SAI.2014.6918206>.
- Bi, X., Smith, B. A., e Zhai, S. (2012). Multilingual touchscreen keyboard design and optimization. *Human-Computer Interaction*, 27(4):352–382. URL <http://www.tandfonline.com/doi/abs/10.1080/07370024.2012.678241>.
- Dell’Amico, M., Díaz, J. C., Iori, M., e Montanari, R. (2009). The single-finger keyboard layout problem. *Computers & Operations Research*, 36(11):3002–3012. ISSN 0305-0548. URL <http://dx.doi.org/10.1016/j.cor.2009.01.018>.
- Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47:381–391. URL <https://www.researchgate.net/publication/10366791>.
- Focacci, F., Laburthe, F., e Lodi, A. (2004). *Local Search and Constraint Programming*, p. 293–329. Springer US, Boston, MA. ISBN 978-1-4419-8917-8. URL https://doi.org/10.1007/978-1-4419-8917-8_9.
- Frieze, A., Yadegar, J., El-Horbaty, S., e Parkinson, D. (1989). Algorithms for assignment problems on an array processor. *Parallel Computing*, 11(2):151 – 162. ISSN 0167-8191. URL <http://www.sciencedirect.com/science/article/pii/0167819189900252>.
- Li, Y., Chen, L., e Goonetilleke, R. S. (2006). A heuristic-based approach to optimize keyboard design for single-finger keying applications. *International Journal of Industrial Ergonomics*, 36(8):695 – 704. ISSN 0169-8141. URL <http://www.sciencedirect.com/science/article/pii/S0169814106000886>.

- Lourenço, H. R., Martin, O. C., e Stützle, T. (2010). *Iterated Local Search: Framework and Applications*, p. 363–397. Springer US, Boston, MA. ISBN 978-1-4419-1665-5. URL https://doi.org/10.1007/978-1-4419-1665-5_12.
- MacKenzie, I. S., Sellen, A., e Buxton, W. A. S. (1991). A comparison of input devices in element pointing and dragging tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '91, p. 161–166, New York, NY, USA. ACM. ISBN 0-89791-383-3. URL <http://doi.acm.org/10.1145/108844.108868>.
- MacKenzie, I. S. e Zhang, S. X. (1999). The design and evaluation of a high-performance soft keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, p. 25–31, New York, NY, USA. ACM. ISBN 0-201-48559-1. URL <http://doi.acm.org/10.1145/302979.302983>.
- MacKenzie, I. S., Zhang, S. X., e Soukoreff, R. W. (1999). Text entry using soft keyboards. *Behaviour & Information Technology*, 18(4):235–244. URL <https://doi.org/10.1080/014492999118995>.
- Ministério da Cultura (2017). Português é hoje a quarta língua mais falada no mundo. <http://www.cultura.gov.br/>. Acessado em: 01-03-2018.
- Mladenović, N. e Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097 – 1100. ISSN 0305-0548. URL <http://www.sciencedirect.com/science/article/pii/S0305054897000312>.
- Osman, I. H. e Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations Research*, 63(5):511–623. ISSN 1572-9338. URL <https://doi.org/10.1007/BF02125421>.
- Sahni, S. e Gonzalez, T. (1976). P-complete approximation problems. *J. ACM*, 23(3):555–565. ISSN 0004-5411. URL <http://doi.acm.org/10.1145/321958.321975>.
- Yamada, H. (1980). A historical study of typewriters and typing methods: from the position of planning japanese parallels. *Journal of Information Processing*, 2(4):175–202. ISSN 1882-6652. URL <https://ci.nii.ac.jp/naid/110002673261/en/>.
- Zhai, S., Hunter, M., e Smith, B. A. (2002). Performance optimization of virtual keyboards. *Human-Computer Interaction*, 17(2-3):229–269. URL <http://www.tandfonline.com/doi/abs/10.1080/07370024.2002.9667315>.