Matemática Contemporânea, Vol. 45, 134–142
http://doi.org/10.21711/231766362017/rmc4515
©2017, Sociedade Brasileira de Matemática

Short Block-Move–CPP is NP-Complete

Luís F. I. Cunha **Vinícius F. dos Santos**
Luis A. B. Kowada **Celina M. H. de Figueiredo**

Abstract

The Closest Object Problem aims to find one object in the center of all others. It was studied for strings with respect to the Hamming distance, where the Hamming Closest String Problem was settled to be NP-complete. The Closest Permutation Problem (CPP) was also studied, since permutations are the natural restrictions of general strings, and we have settled that the Block interchange–CPP and the Breakpoint–CPP are NP-complete.

We consider a restricted form of block-interchange, called short block-move, defined by exchanging two contiguous blocks of elements of total length at most 3, for which the computational complexity of the distance problem is still open. We provide sufficient conditions to determine the short block-move distance by showing that the optimal sorting sequence of short block-moves of a given permutation can be obtained by sorting each connected component separately on the permutation graph, and we prove that Short Block Move–CPP is NP-complete.

2000 AMS Subject Classification: 68Q17, 68P10, 05A05.
Key Words and Phrases: Closest problem, NP-Completeness; Hamming distance; short block move
This research was supported by CNPq, CAPES, and FAPERJ.

1 Introduction

The Closest Permutation Problem (CPP) is a combinatorial challenge with applications in computational biology [4], where an input permutation set models a set of genomes, and we want to find a solution genome that is closely related to all others, i.e. a permutation minimizing the *radius* of the input permutation set. Several metrics corresponding to genome rearrangements such as Cayley, transposition, block-interchange, breakpoint, and reversal have been studied [4]. Popov [7] studied the CPP regarding the Cayley metric and proved that the Cayley–CPP is NP-complete. We have studied the CPP regarding the block interchange and the breakpoint metrics, and proved that the Block interchange–CPP and Breakpoint–CPP are NP-complete [3]. The CPP has not been studied regarding other metrics to compute distances, for instance with respect to short block-move, for which the distance problem is open so far [5].

This extended abstract is organized as follows: in Section 2 we present the definitions of the Hamming Closest String Problem, the Closest Permutation Problem, and the metrics we deal with; in Section 3 we show that an optimal sorting sequence of short block-moves can be found by sorting each connected component separately on the permutation graph; in Section 4 we show that Short Block Move–CPP is NP-complete; in Section 5 we discuss some open questions for further work about complexity of other related metrics.

2 Closest problem and short block-move distance

An *alphabet* Σ is a non empty set of *letters*, and a *string* over Σ is a sequence of letters of Σ. The *Hamming distance between two strings* s_i and σ denoted $d_H(s_i, \sigma)$ is defined as the number of mismatched positions between s_i and σ. We call the *Hamming distance of a string* the number of mismatched positions between s_i and $\iota = 0^m$, such that s_i has length m.

The Hamming Closest String Problem (H–CSP in brief) is defined as follows: given a set $\{s_1, s_2, \ldots, s_\ell\}$ of strings of length m over the alphabet Σ and a non-negative integer f, decide the existence of a string σ of length m such that $\max\limits_{i=1,\ldots,\ell} d_H(s_i, \sigma) \leq f$. A *solution of H–CSP* is any string σ that satisfies $\max\limits_{i=1,\ldots,\ell} d_H(s_i, \sigma) \leq f$. Lanctot *et al.* [6] proved that H–CSP is NP-complete for a binary alphabet.

A *permutation* π of length n is a string with a unique occurrence of each letter, since the function $\pi : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ is a bijection. We define $\pi = [\pi(1)\, \pi(2) \cdots \pi(n)]$.

Given a metric M and $d_M(p_i, \pi)$ the minimum number of operations regarding the metric M to transform p_i into π, the Closest Permutation Problem is defined as follows:

Metric M Closest Permutation Problem (M–CPP)
INPUT: Set of permutations $\{p_1, p_2, \ldots, p_k\}$ of length n and a non-negative integer d.
QUESTION: Is there a permutation π of length n such that $\max\limits_{i=1,\ldots,k} d_M(p_i, \pi) \leq d$?

In case of a positive answer for M–CPP, we call a *solution of M–CPP* any permutation π that satisfies $\max\limits_{i=1,\ldots,k} d_M(p_i, \pi) \leq d$.

Given a set of permutations, the Closest Permutation Problem aims to find a solution permutation that minimizes the maximum distance between the solution and all other input permutations. The metric to compute distances depends on the context of the problem.

We call the *distance between two permutations* the minimum number of the corresponding operations to transform one permutation into another one. The *distance of a permutation* is the minimum number of operations to transform the permutation into the *identity permutation* $\iota = [1\ 2\ \cdots\ n]$.

In this work, we deal with short block-moves, a particular form of block-interchange, which is an operation that transforms one permutation into another one by exchanging two blocks.

Note that regarding the distance problem, general operations do not imply the same complexity with respect to the complexity of more particular operations. For instance, the block-interchange distance can be computed in polynomial time (see [2]), while in contrast, the transposition distance, a block-interchange of two consecutive blocks, is an NP-complete problem as proved in [1]. On the other hand, if a distance problem is NP-complete, then the closest problem for the same operation is also NP-complete.

A *p-bounded block-move* is a transposition for which the sum of the number of elements is bounded by p. The restricted 3-bounded block-move is called a *short block-move*, a problem proposed by Heath and Vergara in [5].

Hence, a short block-move may equivalently be seen as a move of either some element past one element, called a *skip*, or past two elements, called a *hop*. The *short block-move distance* $d_{sbm}(\pi)$ is the minimum number of short block-moves needed to transform π into ι.

Now, we review how to obtain bounds for the SBM distance, based on the permutation graph.

The short block-move distance To estimate the SBM distance, Heath and Vergara in [5] used the *permutation graph* $PG(\pi) = (V_\pi^p, E_\pi^p)$, where $V_\pi^p = \{1,\ 2,\ \ldots, n\}$ and $E_\pi^p = \{(i,j) \mid \pi_i > \pi_j,\ i < j\}$ (that is, every edge of $PG(\pi)$ is an *inversion*). They proved that the number of inversions gives bounds for the distances, since each short block-move decreases the number of inversions by at least one unit, and by at most two units. Hence: $\left\lceil \frac{|E_\pi^p|}{2} \right\rceil \leq d_{sbm}(\pi) \leq |E_\pi^p|$.

Given a permutation, our goal is to minimize the number of operations that decrease only one inversion. Examples of permutations for which the distances achieve the lower and the upper bound are [2 4 3 5 1] and [2 1 4 3], respectively.

A short block-move is a *correcting* move if it eliminates one or two inversions. Otherwise, the short block-move is called *non-correcting*. Heath and Vergara [5] proved that each sorting sequence can be performed by using just correcting moves. Table 1 shows the replacements from non-correcting moves to correcting moves in an optimal sorting sequence (in all cases, $e < f$ and x is arbitrary).

case	π	$\pi' = \pi\beta_i$	$\pi'' = \pi\beta'_i$
1	$\cdots ef \cdots$	$\cdots fe \cdots$	$\cdots ef \cdots$
2	$\cdots exf \cdots$	$\cdots xfe \cdots$	$\cdots xef \cdots$
3	$\cdots exf \cdots$	$\cdots fex \cdots$	$\cdots efx \cdots$
4	$\cdots xef \cdots$	$\cdots fxe \cdots$	$\cdots exf \cdots$
5	$\cdots efx \cdots$	$\cdots fxe \cdots$	$\cdots exf \cdots$

Table 1: How to replace a non-correcting move β_i with a correcting move β'_i (see [5]).

Next, we prove that sorting each connected component of the permutation graph PG separately is an optimal strategy.

3 Sorting connected components separately

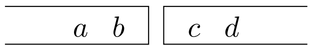
We sometimes use graph-theoretic terminology directly on permutations instead of their permutation graphs. For instance, we say that π is connected (meaning that its permutation graph is), or that a permutation σ is a connected component of a permutation π (meaning that the permutation graph of σ is a connected component of the permutation graph of π).

Let us refer to moves that introduce elements in connected components of the permutation as *merging moves*. For instance, $[2\ 3\ \underline{1\ 6}\ \underline{4}\ 5] \rightarrow [2\ 3\ 4\ 1\ 6\ 5]$ is a merging move.

Theorem 1. *For every permutation π, sorting each connected component of π separately is optimal.*

Proof (sketch). We allow ourselves to use merging moves, which can be replaced by the correcting moves given in Table 1. The modified sequence is not longer than the original, and we observe that these new moves never merge components.

A merging move must act on contiguous components of π. Let us assume that in the leftmost component, the move acts on the ends with elements a and b, and that the component at the right starts with elements c and d.



It implies that $a < c$, $a < d$, $b < c$ and $b < d$. Now, we replace any merging move involving those component's extremities with correcting moves. There are five cases to consider: $\underline{a}\ \underline{b\ c}\ d \to b\ c\ a\ d$, $\underline{a\ b}\ \underline{c}\ d \to c\ a\ b\ d$, $a\ \underline{b}\ \underline{c}\ d \to a\ c\ b\ d$, $a\ \underline{b}\ \underline{c\ d} \to a\ c\ d\ b$ and $a\ \underline{b\ c}\ \underline{d} \to a\ d\ b\ c$.

None of the correcting moves that we use to replace the non-correcting moves in those five cases is a merging move, and no such replacement increases the length of our sorting sequence. Given any sorting sequence, we repeatedly apply the above transformation to the merging move with the smallest index until no such move remains. In particular, the transformation applies to optimal sequences as well, and the proof is complete. □

Note that there exist cases where allowing merging moves still yields an optimal solution. This is the case for [2 1 4 3], which can be optimally sorted as follows: $[2\ \underline{1\ 4}\ \underline{3}] \to [\underline{2\ 3}\ \underline{1}\ 4] \to \iota$.

It is natural to wonder whether Theorem 1 generalizes to p-bounded block-move, for $p > 3$. However, the following counterexample shows that it is not the case, even for a bound of 4: sorting each component of [3 2 1 6 5 4] separately yields a sequence of length four, but one can do better by merging components as follows: $[3\ 2\ \underline{1\ 6}\ \underline{5\ 4}] \to [3\ \underline{2\ 5}\ \underline{4\ 1}\ 6] \to [\underline{3\ 4}\ \underline{1\ 2}\ 5\ 6] \to \iota$. ptimal sorting sequence.

Next, we consider the closest permutation problem. We apply transformations from a generic instance of the H–CSP to particular instances of the SBM–CPP. We establish a relationship between the Hamming distance of binary strings and the permutation distance of the corresponding short block-move distance.

4 Short Block Move–CPP is NP-complete

Firstly, we apply Algorithm 1 that transforms an arbitrary binary string s of length m into a particular permutation λ_s of length $2m$.

```
Algorithm 1: $Permut_{BI}(s)$
  input  : Binary string $s$ of length $m$
  output: Permutation $\lambda_s$
1 Each occurrence of 0 in position $i$ corresponds to the elements
    $2i-1$ and $2i$ in positions $2i-1$ and $2i$, respectively.
2 Each occurrence of 1 in position $i$ corresponds to the elements
    $2i-1$ and $2i$ in positions $2i$ and $2i-1$, respectively.
```

Next, we establish the key equality between the Hamming distance of an input string s and the short block-move distance of its output permutation λ_s obtained from Algorithm 1.

Lemma 1. *Given a string $s \neq 0^m$ of length m and a permutation λ_s of length $2m$ obtained by Algorithm 1, the short block-move distance of λ_s is $d_{sbm}(\lambda_s) = d_H(s)$.*

Proof. By Theorem 1, each connected component can be sorted separately and each bit 1 corresponds exactly to an inversion. □

Now, we show how a solution for the H–CSP implies a solution for the Short block move–CPP, and vice versa.

Lemma 2. *Given a set of k permutations obtained by Algorithm 1, there is a Short block move closest permutation with maximum distance at most d if and only if there is a Hamming closest string with maximum distance equal to d.*

Proof (sketch). ($\Rightarrow$) Given a solution permutation λ', if λ' can be built by Algorithm 1 for some input string s', then, by Lemma 1, s' is a closest string. Otherwise, we search the permutation from the left to the right in order to find the first position i where the corresponding element is distinct from the one given by Algorithm 1. After, we continue searching to the right until we find an element j agreeing with the algorithm output to be placed at position i. Therefore, we obtain a new permutation with a longer prefix agreeing with the algorithm output by placing such element j at position i, without increasing the distance to any input permutation, since we do not increase the number of inversions. Repeating this procedure, a string agreeing with the algorithm output can be found. By Lemma 1, a string of maximum distance equal to d can be constructed.

($\Leftarrow$) Given a solution string s, we obtain the associated permutation λ_s given by Algorithm 1. By Lemma 1, we have the solution s regarding the H–CSP corresponding to the permutation λ_s with the same value of maximum distance d. □

Theorem 2. *The* Short Block Move–CPP *is* NP*-complete.*

5 Further work

Note that the Short Block Move–CPP is NP-complete regarding other instances. An example of this fact is obtained by associating each bit 0 to the identity permutation and each bit 1 to any permutation π such that its short block-move distance is known. In this case, we generalize the result of Lemma 1 to the equality $d_{sbm}(\lambda_s) = d_{sbm}(\pi)d_H(s)$.

Since we show that the Short Block Move–CPP is NP-complete, the following question arises: is the p-Bounded Block Move–CPP NP-complete? As we discussed in Section 3, Theorem 1 does not hold for p-bounded block moves, for $p \geq 4$. Consequently, new ideas are necessary to establish the complexity of p-bounded Block Move–CPP.

References

[1] Bulteau, L., Fertin, G., Rusu, I. Sorting by transpositions is difficult. SIAM J. Disc. Math, 26(3), 1148–1180 (2012)

[2] Christie, D. A.: Sorting permutations by block-interchange. Inf. Process. Lett., 60, 165–169 (1996)

[3] Cunha L. F. I., dos Santos V. F. S., Kowada, L. A. B., de Figueiredo, C. M. H.: The block-interchange and the breakpoint CLOSEST PERMUTATION problems are NP-Complete, in Proceedings of the 18th Latin-Iberoamerican Conference on Operations Research, 239–246 (2016).

[4] Fertin, G., Labarre, A., Rusu, I., Tannier, E., Vialette, S.: Combinatorics of Genome Rearrangements. The MIT Press (2009)

[5] Heath, L. S. and Vergara, J. P. C.: Sorting by short block-moves. Algorithmica, 28, pp. 323–354, (2000)

[6] Lanctot, J. K., Li, M., Ma, B., Wang, S., Zhang, L.: Distinguishing string selection problems. Inform. and Comput., 185(1), 41–55 (2003)

[7] Popov, V. Y.: Multiple genome rearrangement by swaps and by element duplications. Theor. Comput. Sci., 385, 115–126 (2007)

Luís F. I. Cunha
UFRJ, Rio de Janeiro, Brazil
UFF, Niterói, Brazil
lfignacio@ic.uff.br

Vinícius F. dos Santos
UFMG, Minas Gerais, Brazil
viniciussantos@dcc.ufmg.br

Luís A. B. Kowada
UFF, Niterói, Brazil
luis@ic.uff.br

Celina M. H. de Figueiredo
UFRJ, Rio de Janeiro, Brazil
celina@cos.ufrj.br