Assembling a New and Improved Transposition Distance Database

Jamile Gonçalves, **Letícia R. Bueno**, **Rodrigo A. Hausen**
CMCC, Universidade Federal do ABC (UFABC)
Santo André – SP – Brazil
jamile.goncalves@aluno.ufabc.edu.br, leticia.bueno@ufabc.edu.br, hausen@compscinet.org

ABSTRACT

Determining the transposition distance of permutations is NP-hard, but the problem of the transposition diameter is still open and it is known only for $n \leq 15$. Previously, every diametral permutation for $n \leq 13$ was found, but only a few instances have been described for $n = 14$ and 15. In this work, we calculate the transposition distance of every permutation for $n \leq 14$ via a breadth-first search on the Cayley graph of permutations of n elements, using toric equivalences to reduce the search space. As a consequence, we have determined and identified all diametral permutations for $n = 14$. We also provide a look-up table having the distance of every permutation of up to 14 elements.

KEYWORDS. Bioinformatics, Genome rearrangement, Transposition diameter.

Main area: OC - Combinatorial Optimization

1. Introduction

Recent technological advances allowed the extraction of a large amount of information about the molecular biology of organisms. This availability has given rise to many methods for genome comparison, most of which aim to produce a number – the distance – that expresses how closely related two organisms are. The distance between two species is specially relevant in the study of the evolution of the species and for the reconstruction of phylogenetic trees.

Rearrangement of large portions of a genome is common [Nadeau and Taylor(1984), Palmer and Herbon(1988)]. Therefore, one method used to determine the distance between genomes of two different species compares large portions of their genomes, and it determines the distance between them by applying a series of successive rearrangements to the order of the blocks of genes of the first genome until the second one is obtained. This distance is called *rearrangement distance* and consists of the minimum number of mutations needed to transform a genome into another. The reason for adopting the minimum number of mutations comes from the *parsimony hypothesis*, which assumes that the most parsimonious scenario is the one that requires the least amount of changes.

In the genome model we will consider in this paper, a chromosome π with n genes is represented by a permutation $\pi = [\pi_1 \pi_2 \dots \pi_n]$ of integers between 1 and n, where each element π_i represents a gene. We assume that there are not repeated genes, therefore the permutation π does not have repeated elements as well. The permutation $[n\ n-1 \dots 2\ 1]$ is called the *reverse permutation* and the permutation $[1\ 2 \dots n-1\ n]$ is called the *identity permutation*.

Distinct rearrangement events affect portions of genes in different ways. One of these events is the *transposition*, which is an operation that moves a block of genes from a region to another inside the same chromosome. The biological meaning for a transposition is a duplication of a block of genes followed by the removal of the original block [Boore(2000)]. Figure 1 shows the transformation of the reverse permutation into the identity permutation for $n = 5$ using transpositions.

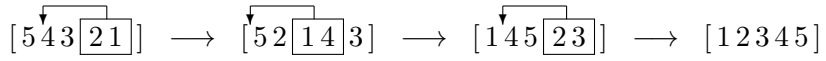


Figure 1: One way to transform the reverse permutation of 5 elements into the identity permutation by transpositions.

Given a permutation π, the *transposition distance problem* consists of determining the minimum number of transpositions needed to transform π into the identity permutation. The *transposition distance* of π, denoted by $d(\pi)$, is a relevant evolutionary measure [Boore(2000), Sankoff et al.(1992)] and several approximation and heuristic algorithms have been proposed for this problem [Bafna and Pevzner(1995), Benoît-Gagné and Hamel(2007), Elias and Hartman(2006), Feng and Zhu(2007), Gu et al.(1999), Hartman and Shamir(2006)]. Other works have focused in estimating bounds for the transposition distance of a permutation [Christie(1999), Elias and Hartman(2006), Eriksson et al.(2001), Hausen et al.(2010), Labarre(2006)]. However, tight bounds for the transposition distance remain to be found in the general case, and only recently it was proved that the problem is NP-hard [Bulteau et al.(2011)].

The largest transposition distance of any permutation of n elements is called the *transposition diameter* and is denoted by $D(n)$. The values of the transposition diameter are determined only for $n \leq 15$ and $n = 17$ [Sloane(2013), Eriksson et al.(2001), Meidanis et al.(1997)]. For $n = 16$ and $n > 17$, [Meidanis et al.(1997)] and [Elias and Hartman(2006)] provide the best known lower bounds, respectively $\lfloor n/2 \rfloor + 1$ for n even and $\lfloor n/2 \rfloor + 2$ for n odd; for the upper bound, $\lfloor (2n-2)/3 \rfloor$ is currently the best one [Eriksson et al.(2001)]. However, tight bounds are not known yet and the computational complexity of determining the transposition diameter remains open.

The difficulty of establishing the transposition diameter is shown in the number of conjectures and in results that were, later, shown to be incorrect. It is known [Meidanis et al.(1997)] that the distance of the reverse permutation is $\lfloor n/2 \rfloor + 1$ for $n > 2$, and that was conjectured as the transposition diameter. This conjecture was later proven invalid [Eriksson et al.(2001)] in the general case, for there are permutations for $n = 13$ and $n = 15$ whose distance is $\lfloor n/2 \rfloor + 2$. From these permutations, [Elias and Hartman(2006)] showed how to construct permutations for n odd whose distance is $\lfloor n/2 \rfloor + 2$. Later, [Lu and Yang(2010)] postulated that, for large values of n, there are permutations – called *super-bad permutations* – whose transposition distance is greater than the distance of the reverse permutation. However, [Cunha et al.(2012)] showed that there are no super-bad permutations. The Table 1 summarizes the results about the transposition diameter.

A permutation π of n elements is called *diametral* if $d(\pi) = D(n)$. Every diametral permutation for $n \leq 13$ is known [Meidanis et al.(1997), Eriksson et al.(2001), Galvão and Dias(2011b)], but only a few instances have been described for $n \geq 14$ [Eriksson et al.(2001), Elias and Hartman(2006)]. In this work, we calculate the transposition distance of every permutation for $n \leq 14$ via a breadth-first search on the Cayley graph of permutations of n elements, using toric equivalences to reduce the search space. As a consequence, we have identified all diametral permutations for $n = 14$.

In Section 2., we give some definitions used in our algorithm, which is presented in Section 3. Computational results are discussed in Section 4. Finally, some conclusions and open problems are discussed in Section 5.

Table 1: Known values and bounds for the transposition diameter

n	**Transposition Diameter** $D(n)$	**References**
$3 \leq n \leq 12$	$D(n) = \lfloor n/2 \rfloor + 1$	[Bafna and Pevzner(1995)] [Meidanis et al.(1997)]
13	$D(n) = \lfloor n/2 \rfloor + 2$	[Eriksson et al.(2001)]
14	$D(n) = \lfloor n/2 \rfloor + 1$	[Eriksson et al.(2001)]
15	$D(n) = \lfloor n/2 \rfloor + 2$	[Eriksson et al.(2001)]
$n \geq 16$, even	$\lfloor n/2 \rfloor + 1 \leq D(n) \leq \lfloor (2n-2)/3 \rfloor$	[Meidanis et al.(1997)] [Eriksson et al.(2001)]
17	$D(n) = \lfloor n/2 \rfloor + 2$	[Elias and Hartman(2006)]
$n \geq 17$, odd	$\lfloor n/2 \rfloor + 2 \leq D(n) \leq \lfloor (2n-2)/3 \rfloor$	[Elias and Hartman(2006)] [Eriksson et al.(2001)]

2. Preliminaries

In this section we introduce the definitions and results that will be used in the construction of our algorithm.

2.1. Mathematical formalization

Let n be a fixed integer. A *transposition* $t(i, j, k)$, where $1 \leq i < j < k \leq n + 1$, is the permutation $[1\ 2 \ldots i{-}1\ j\ j{+}1 \ldots k{-}1\ i\ i{+}1 \ldots j{-}1\ k\ k{+}1 \ldots n]$ — let the reader beware: in group theory, the name "transposition" may also be used with another definition that differs from ours.. A sequence of transpositions $t(i_1, j_1, k_1), t(i_2, j_2, k_2) \ldots, t(i_\ell, j_\ell, k_\ell)$ *sorts* a permutation π if $\pi \cdot t(i_1, j_1, k_1) \cdot t(i_2, j_2, k_2) \cdots t(i_\ell, j_\ell, k_\ell) = \iota$, where the product "$\cdot$" denotes the composition of permutations as an action to the right.

The *transposition distance* of π, denoted $d(\pi)$, is defined thusly:

$$d(\pi) := \min\{\ell; \pi \cdot t(i_1, j_1, k_1) \cdot t(i_2, j_2, k_2) \cdots t(i_\ell, j_\ell, k_\ell) = \iota\}.$$

The *transposition diameter* $D(n)$ is the maximum transposition distance attained by a permutation of n elements, i.e.,

$$D(n) := \max\{d(\pi); \pi \text{ is a permutation of } n \text{ elements}\}.$$

2.2. The Cayley graph of transpositions

The *Cayley graph of transpositions* [Eriksson et al.(2001)], also known as the *transposition rearrangement graph* [Hausen et al.(2010)] and denoted by $TRG(n)$, is an undirected graph whose vertex set is the set of all permutations of n elements. There is an edge between π and σ in $TRG(n)$ if, and only if, $\pi \cdot t(i, j, k) = \sigma$ for some transposition $t(i, j, k)$. Figure 2 depicts $TRG(3)$ and $TRG(4)$. It follows immediately from the definition of $TRG(n)$ that:

- a path between π and ι in $TRG(n)$ corresponds to a sequence of transpositions that sorts π;
- the length of a shortest path between π and ι in $TRG(n)$ is $d(\pi)$; and
- the diameter (greatest length of a shortest path between any two vertices) of $TRG(n)$ is $D(n)$.

Therefore, both the transposition distance $d(\pi)$ of a given permutation π and the transposition diameter $D(n)$ can be obtained by a breadth-first search (BFS) on $TRG(n)$

that begins at ι. Unfortunately, the apparent simplicity of finding these two parameters in $TRG(n)$ is dispelled by the size of this graph – it has $n!$ vertices and $\frac{n!(n^3-n)}{12}$ edges [Hausen et al.(2010)]. Although there is not a known manner of searching $TRG(n)$ in an efficient way, i.e., one whose space and time complexity are at least polynomial in n, in Section 2.3. we present the reduction of the Cayley graph that allowed us to obtain the results in this paper.

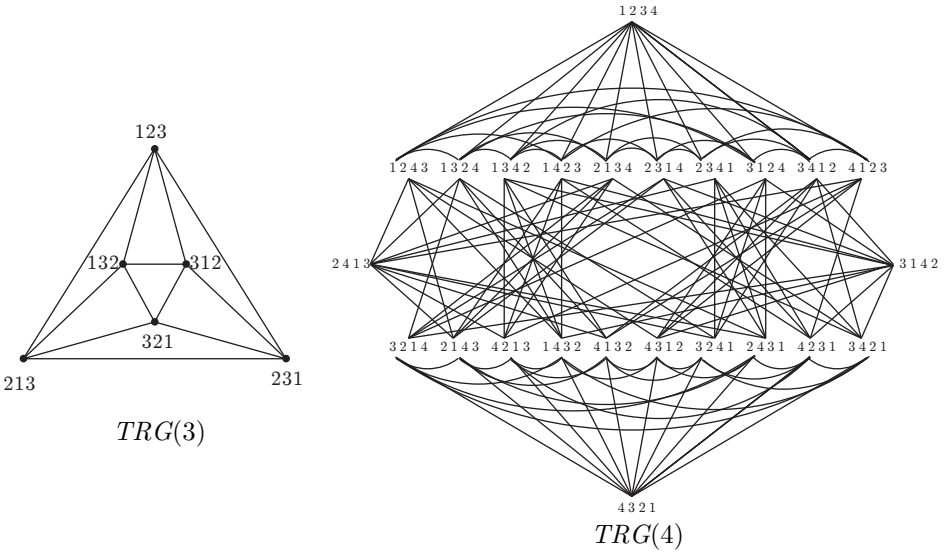


Figure 2: The Cayley graphs $TRG(3)$ and $TRG(4)$.

2.3. Reduction of a Cayley graph via toric classes

In order to show how a reduction of the Cayley graph is obtained, we must first lay out some results. Given a permutation π and an integer x, the *cyclic shift* $\pi + x$ is defined [Eriksson et al.(2001)] as the permutation obtained by the following steps:

i) consider $\pi = [\pi_1\pi_2 \dots \pi_n]$ and $\pi_0 = 0$;
ii) for $i = 0 \dots n$, calculate $\pi_i + x \pmod{n+1}$, i.e., the remainder of the division of $\pi_i + x$ by $n+1$;
iii) let $\pi + k = [\gamma_1 \dots \gamma_n]$, where $\gamma_i = \pi_\ell + x$, for $\ell = j + i \pmod{n+1}$, and j is such that $0 = \pi_j + x \pmod{n+1}$.

Example 1 *Calculate* $[132] + 2$*:*

i) consider π_0*,* π_1*,* π_2 *and* π_3 *respectively equal to* 0*,* 1*,* 3 *and* 2*;*
ii) calculating $\pi_i + 2$ *(*mod 4*), we obtain* 2*,* 3*,* 1 *and* 0*, and;*
iii) since $\pi_3 + 2 = 0$*, we have* $j = 3$*,* $\gamma_1 = \pi_0 + 2 = 2$*,* $\gamma_2 = \pi_1 + 2 = 3$*, and* $\gamma_3 = \pi_2 + 2x = 1$*, therefore* $[132] + 2 = [231]$.

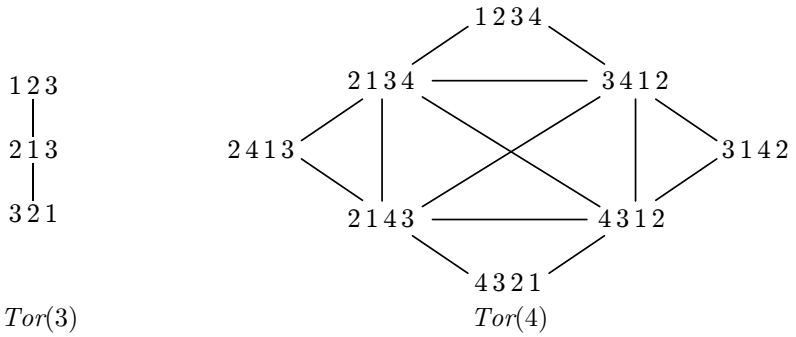
Two permutations π and σ are said to belong to the same *toric class* if $\sigma = \pi + x$ for some integer x such that $1 \leq x \leq n$. Since $[132] + 2 = [231]$, as in Example 1, both permutations are in the same toric class.

Notice that $\iota + x = \iota$ for every n and x, and that $d(\pi) = d(\pi + x)$, since these cyclic shifts just relabel π and ι in the same manner. Therefore, Theorem 1 is valid.

Theorem 1 *[Eriksson et al.(2001)] If two permutations are in the same toric class, they have the same distance to the identity.*

A toric class has at least one permutation (for instance, the toric class that contains ι) and at most $n+1$ permutations (such as the toric class in which [132], [312], [231] and [213] belong). The number of elements in a toric class is always a divisor of $n+1$, and there are exactly $\varphi(n+1)$ classes that have only one element [Christie(1999), Hausen et al.(2010)], where φ is Euler's totient function.

The *toric graph* $Tor(n)$ is a graph in which every vertex is a permutation of n elements that represents a toric class, and two vertices π, σ are adjacent if, and only if, there are two integers x and y such that $\pi + x$ and $\sigma + y$ are adjacent in $TRG(n)$. Figure 3 depicts the toric graphs $Tor(3)$ and $Tor(4)$.



$Tor(3)$ $Tor(4)$

Figure 3: The toric graphs $Tor(3)$ and $Tor(4)$.

Notice that the length of any shortest path from a permutation π to ι is the same in $TRG(n)$ and in $Tor(n)$, but $Tor(n)$ has less vertices and edges. Therefore, the calculation of distances via BFS is faster using the toric graph.

2.4. Reductions of permutations

Given a permutation $\pi = [\pi_1\, \pi_2 \ldots \pi_n]$, and considering $\pi_0 = 0$ and $\pi_{n+1} = n+1$, an *adjacency* in π is a pair of consecutive elements π_i, π_{i+1}, where $0 \leq i \leq n$, such that $\pi_{i+1} = \pi_i + 1$. If two consecutive elements are not an adjacency, they are a *breakpoint*. The number of breakpoints in π is denoted by $b(\pi)$. It follows that $b(\pi) = 0$ if, and only if, $\pi = \iota$.

Example 2 *Let* $\pi = [3\,4\,1\,2]$. *The breakpoints are the pairs:* $0,3$ *;* $4,1$ *;* $2,5$. *The adjacencies are:* $3,4$ *and* $1,2$.

Notice that it is always possible to sort any permutation without breaking any adjacencies. Hence, a block of two or more consecutive elements that only have adjacencies among them can be thought of as only one indivisible element. Given a permutation π, the *reduced permutation* $gl(\pi)$ is obtained from π by replacing every block of adjacent elements with the smallest element in that block, and relabeling the elements according to their lexicographic order [Christie(1999)].

Example 3 *Considering* $\pi = [3\,4\,1\,2]$, *we replace the adjacency* $3,4$ *with* 3, *and* $1,2$ *with* 1, *obtaining* $[31]$. *Relabeling* $[31]$, *we have* $gl(\pi) = [21]$.

Observe that, since there is no need to break adjacencies when sorting a transposition, π and $gl(\pi)$ always have the same distance. This property is used in Algorithm 3.

3. An Algorithm for the Transposition Diameter

Algorithm 1 calculates the transposition distance of every permutation of n elements via a BFS on $Tor(n)$ starting at ι. Since it is currently impossible to store in memory an

explicit representation of a toric graph where $n > 13$, we use two main data structures: i) an array of $n!$ nibbles (half byte) that serves the double purpose of marking the visited permutations (along with their torically equivalent permutations, which are calculated in the underlined lines in the Algorithm) and storing their respective distances; and ii) a queue to store the ranks of the permutations that are going to be visited next. The edges are dynamically calculated in the line that reads "for all transposition $t\ldots$". In the end, the array of distances is saved to a binary file, to be used in Algorithms 2 and 3.

Algorithm 1 *(Calculates the transposition distance of every permutation of n elements)*

Input: number of elements n
Output: binary file "*dist***n**.*bin*" with all transposition distances

```
let π be a permutation
let q be an empty queue
let "distn.bin" be a binary file
let d[0...n! − 1] be an array of integers
for all permutation π with n elements do
  d[rank(π)] ← ∞
π ← [1 2 3 ... n]
d[rank(π)] ← 0
enqueue π in q
visited(π) ← true
while q not empty do
  π ← first element of q
  dequeue the first element of q
  for all transposition t such that d[rank(π · t)] = ∞ do
    d[rank(π · t)] ← d[rank(π)] + 1
    k ← 1
    while (π · t) + k is different from π · t do
      d[rank((π · t) + k)] ← d[rank(π · t)]
      k ← k + 1
    enqueue π·t in the end of q
save the array d in "distn.bin"
```

The binary files produced by Algorithm 1 serve as a look-up table for an oracle for the transposition distance problem. The oracle loads the files to the main memory (Algorithm 2), putting their data in a matrix form – or, rather, in the form of an array of arrays. With a relatively simple look-up (Algorithm 3), it is possible to obtain the distance of a permutation π for which $d(\pi)$ or $d(gl(\pi))$ has already been computed before. If the database does not have information about the permutation, which only occurs if the permutation cannot be reduced to a permutation having 14 or a smaller number of elements, the oracle returns "undetermined," and it is up to the user of these algorithms to decide what to do next.

4. Computational Results

We executed Algorithm 1 in one core of an Altix 4700 supercomputer (1.4GHz Intel Itanium processor, 257GBytes of main memory) to compute the transposition distances for all permutations of up to 14 elements. For $n = 8\ldots14$, Table 2 shows runtime,

Algorithm 2 *(Loads the distances to main memory)*

Input: integer N that is the greater n for which all distances were calculated; directory *dir*, where are the distance files
Output: matrix $N \times N!$, the oracle
  let $m[1 \ldots N][0 \ldots N! - 1]$ **be** a matrix of integers
  let $d[0 \ldots N! - 1]$ **be** an array of integers
  for all $n \leftarrow 1 \ldots N$ **do**
    read array d do file *"dist**n**.bin"* in the directory *dir*
    for all $r \leftarrow 0 \ldots n! - 1$ **do**
      $m[n][r] \leftarrow d[r]$
  return m

Algorithm 3 *(Queries the look-up tables)*

Input: permutation π; matrix of distances $m[1 \ldots N][0 \ldots N! - 1]$
Output: transposition distance $d(\pi)$ or *"undetermined"* if the distance is not known
  $n \leftarrow$ *number of elements of* π
  if $n \leq N$ **then**
    return $m[n][rank(\pi)]$
  else
    $\pi' \leftarrow gl(\pi)$
    $n \leftarrow$ *number of elements of* π'
    if $n \leq N$ **then**
      return $m[n][rank(\pi')]$
    else
      return *"undetermined"*

maximum number Q of permutations in the queue and the maximum memory usage in bytes, considering $n!/2$ bytes for the array of distances and $8Q$ bytes for the queue.

Table 2: Maximum size of the queue and the runtime of the Algorithm 1

n	**runtime**	**maximum queue size (no. of permutations)**	**maximum memory usage (bytes)**
8	0, 38s	2869	43k
9	4, 54s	23366	360k
10	58, 99s	203951	3.3M
11	5m 33, 42s	2087612	35M
12	1h 26m 43.77s	22752393	402M
13	1d 6h 42m	276323066	5G
14	21d	3612427772	67.6G

We found the distribution of distances for $n \leq 14$ (Tables 3 and 4). The distributions for $n \leq 13$ are identical to the distributions in [Galvão and Dias(2011b)]. This is the first time the distribution of distances for $n = 14$ is computed.

Table 3: Transposition distance distributions for permutations of $n \leq 12$

d	n											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
0	1	1	1	1	1	1	1	1	1	1	1	1
1		1	4	10	20	35	56	84	120	165	220	286
2			1	12	68	259	770	1932	4284	8646	16203	28600
3				1	31	380	2700	13467	52512	170907	484440	1231230
4						45	1513	22000	191636	1183457	5706464	22822293
5								2836	114327	2010571	21171518	157499810
6										255053	12537954	265819779
7												31599601

Table 4: Transposition distance distributions for permutations of $n = 13, 14$

d	n	
	13	**14**
0	1	1
1	364	455
2	48048	77441
3	2864719	6196333
4	78829491	241943403
5	910047453	4334283646
6	3341572727	29432517384
7	1893657570	47916472532
8	427	5246800005

5. Conclusions and Future Works

Determining the transposition distance of permutations is a NP-hard problem. On the other hand, the transposition diameter problem is still open and it is known only for $n \leq 15$ [Sloane(2013)].

In this paper, we computed the transposition distance for all permutations for $n \leq 14$ using Algorithm 1. As a consequence, we have determined and identified all diametral permutations for $n = 14$ in $O(n)$ time or, if the rank of the permutation is already known, in $O(1)$ time.

Previously, every diametral permutation for a given number of elements n has been determined only for n up to 13 [Galvão and Dias(2011b)], and only a handful of examples were known for $n = 14$ [Eriksson et al.(2001)]. We have managed to extend these results for n up to 14 with the use of toric equivalences to reduce the search space, whereas the approach by Galvão and Dias was a more direct implementation of a breadth first search on a Cayley graph [Galvão and Dias(2011a)]. Our approach also differs from Eriksson *et al.*'s approach, in that they have not listed every diametral permutation, proving instead that the distance of a permutation of 14 elements cannot exceed that of the reverse permutation.

From the execution for $n = 8 \ldots 14$, we can estimate the resources needed for $n = 15$ (Figures 4 and 5) considering the exponential tendency for the curves in both graphics. For $n = 15$, we would have a runtime of approximately 6 months and a queue

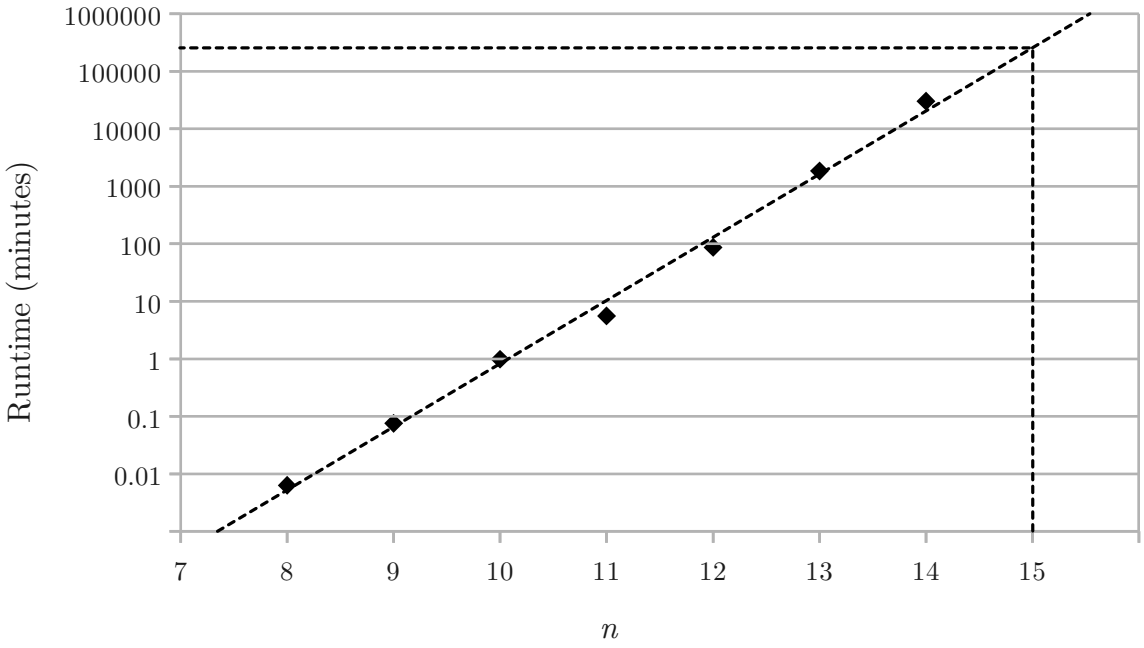


Figure 4: Runtime for Algorithm 1 (log scale for the vertical axis).

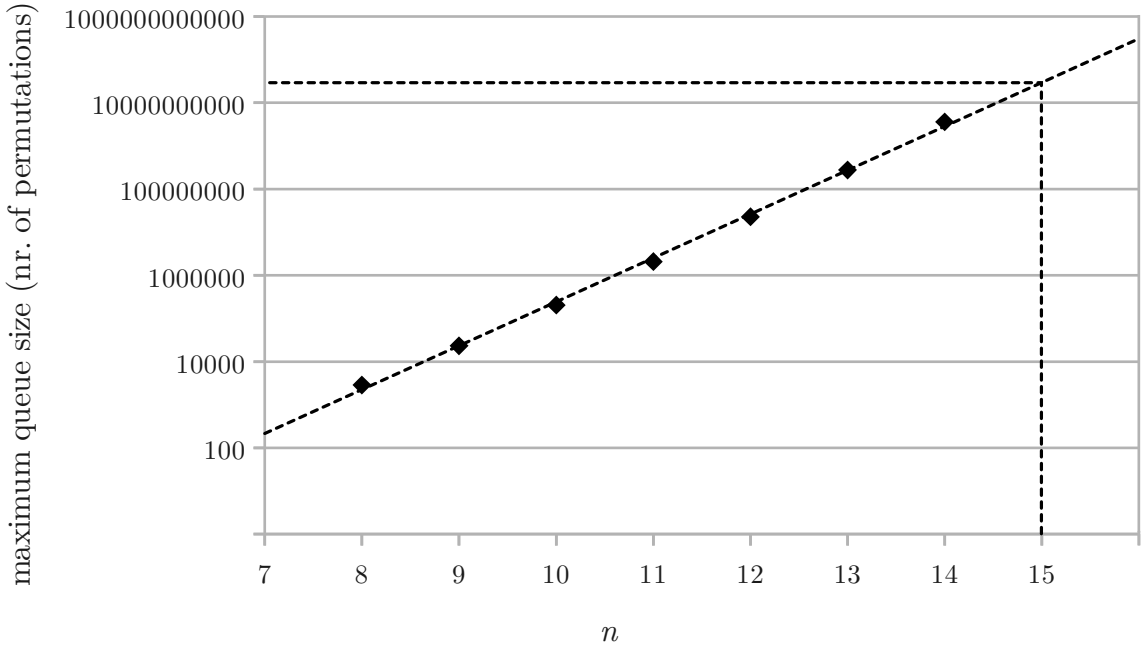


Figure 5: Maximum queue size for Algorithm 1 (log scale for the vertical axis).

of maximum size of roughly 30×10^9 permutations. Since each permutation occupies 8 bytes, the maximum size of the queue would be approximately 224Gigabytes. The distance array would consume 15!/2 bytes, or 609Gbytes. Henceforth, the total amount of memory needed to run Algorithm 1 for $n = 15$ is in the neighborhood of 833Gbytes. We conclude that determining the diametral permutations for $n = 15$ via BFS is not feasible using the current computers, but may become attainable in the next few years.

Aside from the theoretical aspect, one practical use for the data obtained in this study is as a look-up table for speeding-up algorithms for calculating the transposition distance. We showed how the distance database can be used in an oracle for the transposition distance problem (Algorithms 2 and 3). A sample implementation is available at `http://compscinet.org/research/tdd/`.

We intend to use these results as a starting point to search for diametral permutations of 15 and 16 elements, first generating candidates from the diametral permutations of 14 elements, and then using an exact algorithm for the distance, aided by our database, for winnowing the permutations whose distance is 9 or 10. This strategy is embarrassingly parallel and will definitely establish the value of the transposition diameter $D(16)$. Another future development is the improvement of the library so that some queries may be performed directly on disk, reducing the main memory footprint at the expense of the look-up time; for instance, should only the distances for $n \leq 13$ be loaded into main memory, they would consume 3.15 Gbytes, well within the limitations of current desktop computers.

Acknowledgment

The authors wish to thank Fundação Universidade Federal do ABC/CNPq for the undergraduate research grant that funded this research (PIBIC-AF).

References

Bafna, V. and Pevzner, P. (1995). Sorting permutations by tanspositions. In *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, SODA '95, pages 614–623, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Benoît-Gagné, M. and Hamel, S. (2007). A New and Faster Method of Sorting by Transpositions Combinatorial Pattern Matching. In Ma, B. and Zhang, K., editors, *Combinatorial Pattern Matching*, volume 4580 of *Lecture Notes in Computer Science*, chapter 15, pages 131–141. Springer Berlin / Heidelberg, Berlin, Heidelberg.

Boore, J. L. (2000). The duplication/random loss model for gene rearrangement exemplified by mitochondrial genomes of deuterostome animals. In Sankoff, D. and Nadeau, J. H., editors, *Comparative Genomics*, pages 133–148. Kluwer Academic Publishers.

Bulteau, L., Fertin, G., and Rusu, I. (2011). Sorting by transpositions is difficult. In *Proceedings of the 38th international colloquim conference on Automata, languages and programming - Volume Part I*, ICALP'11, pages 654–665, Berlin, Heidelberg. Springer-Verlag.

Christie, D. A. (1999). *Genome Rearrangement Problems*. PhD thesis, University of Glasgow.

Cunha, L. F. I., Kowada, L. A. B., de A. Hausen, R., and de Figueiredo, C. M. H. (2012). Transposition diameter and lonely permutations. In de Souto, M. C. P. and Kann, M. G., editors, *BSB*, volume 7409 of *Lecture Notes in Computer Science*, pages 1–12. Springer.

Elias, I. and Hartman, T. (2006). A 1.375-approximation algorithm for sorting by transpositions. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4):369–379.

Eriksson, H., Eriksson, K., Karlander, J., Svensson, L., and Wästlund, J. (2001). Sorting a bridge hand. *Discrete Mathemathics*, 241(1):289–300.

Feng, J. and Zhu, D. (2007). Faster algorithms for sorting by transpositions and sorting by block interchanges. *ACM Transactions on Algorithms*, 3(3).

Galvão, G. R. and Dias, Z. (2011a). Computing rearrangement distance of every permutation in the symmetric group. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 106–107, New York, NY, USA. ACM.

Galvão, G. R. and Dias, Z. (2011b). On the distribution of rearrangement distances. In *BSB 2011 Digital Proceedings*, pages 41–48, Brasília, Brazil.

Gu, Q.-P., Peng, S., and Chen, Q. M. (1999). Sorting permutations and its applications in genome analysis. *Lectures on Mathematics in the Life Science*, 26:191–201.

Hartman, T. and Shamir, R. (2006). A simpler and faster 1.5-approximation algorithm for sorting by transpositions. *Information and Computation*, 204(2):275–290.

Hausen, R. d. A., Faria, L., Figueiredo, C. M. H. d., and Kowada, L. A. B. (2010). Unitary toric classes, the reality and desire diagram, and sorting by transpositions. *SIAM J. Discrete Math.*, 24(3):792–807.

Labarre, A. (2006). New bounds and tractable instances for the transposition distance. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4):380–394.

Lu, L. and Yang, Y. (2010). A lower bound on the transposition diameter. *SIAM Journal on Discrete Mathematics*, 24(4):1242–1249.

Meidanis, J., Walter, M. E. M. T., and Dias, Z. (1997). Transposition distance between a permutation and its reverse. In Baeza-Yates, R., editor, *Proceedings of 4th South American Workshop on String Processing*, pages 70–79. Carleton University Press.

Nadeau, J. H. and Taylor, B. A. (1984). Lengths of chromosomal segments conserved since divergence of man and mouse. *Proceedings of the National Academy of Sciences of the United States of America*, 81(3):814–818.

Palmer, J. D. and Herbon, L. A. (1988). Plant mitochondrial DNA evolves rapidly in structure, but slowly in sequence. *Journal of Molecular Evolution*, 28(1–2):87–97.

Sankoff, D., Leduc, G., Antoine, N., Paquin, B., Lang, B. F., and Cedergren, R. (1992). Gene order comparisons for phylogenetic inference: evolution of the mitochondrial genome. *Procedings of the National Academy of Sciences*, 89(14):6575–6579.

Sloane, N. J. A. (2013). The on-line encyclopedia of integer sequences: Sequence a065603. http://oeis.org/a065603.