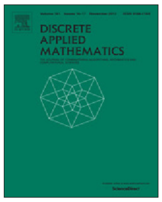
Contents lists available at ScienceDirect

Discrete Applied Mathematics

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



Computing similarity distances between rankings[☆]

Farzad Farnoud (Hassanzadeh) [a,b], Olgica Milenkovic [c], Gregory J. Puleo [d,*], Lili Su [c]

[a] *Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA, USA*
[b] *Department of Computer Science, University of Virginia, Charlottesville, VA, USA*
[c] *Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, USA*
[d] *Coordinated Science Lab, University of Illinois at Urbana-Champaign, Urbana, IL, USA*

ARTICLE INFO

Article history:
Received 19 April 2016
Received in revised form 25 July 2017
Accepted 28 July 2017
Available online xxxx

Keywords:
Permutations
Similarity distance
Transposition distance

ABSTRACT

We address the problem of computing distances between permutations that take into account similarities between elements of the ground set dictated by a graph. The problem may be summarized as follows: Given two permutations and a positive cost function on transpositions that depends on the similarity of the elements involved, find a smallest cost sequence of transpositions that converts one permutation into another. Our focus is on costs that may be described via special metric-tree structures. The presented results include a linear-time algorithm for finding a minimum cost decomposition for simple cycles and a linear-time 4/3-approximation algorithm for permutations that contain multiple cycles. The proposed methods rely on investigating a newly introduced balancing property of cycles embedded in trees, cycle-merging methods, and shortest path optimization techniques.

1. Introduction

The problem of sorting distinct elements according to a given set of criteria has a long history and has been studied in mathematics, computer science, and social choice theory alike [8,11,19]. One volume of the classical text in computer science – Knuth's *The Art of Computer Programming* – is almost entirely devoted to the study of sorting. The solution to the problem is well known when the sorting steps are swaps (transpositions) of two elements: In this case, it is convenient to first perform a cycle decomposition of the permutation and then swap elements in the same cycle until all cycles have unit length.

Sorting problems naturally introduce the need for studying *distances* between permutations. There are many different forms of distance functions on permutations, with the two most frequently used being the Cayley distance and the Kendall distance [5]. Although many generalizations of the Cayley, Kendall and other distances are known [16], only a handful of results pertain to distances in which one assigns positive weights or random costs[1] to the basic rearrangement steps [1,6,14]. Most such work has been performed in connection with genome rearrangement studies [2,7] and for the purpose of gene prioritization [18]. (Note that [2,7] use a different notion of "transposition" than is used in this paper.) Some other examples appear in the social sciences literature (see references in [6]), pertaining to constrained vote aggregation and logistics [12].

[☆] This work was supported in part by the NSF STC Class 2010 CCF 0939370 grant. Research of the third author is supported by the IC Postdoctoral Research Fellowship (grant number 2014-14081100005). Farzad Farnoud was with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, USA.

[*] Corresponding author.
E-mail address: gjp0007@auburn.edu (G.J. Puleo).

[1] Throughout the paper, we use the words cost and weight interchangeably, depending on the context of the exposition.

http://dx.doi.org/10.1016/j.dam.2017.07.038
0166-218X/© 2017 Elsevier B.V. All rights reserved.

A related line of work is the study of sorting algorithms in which *comparisons* between different objects may have different costs depending on the objects in question. Such problems were studied (in a broader context) by Charikar et al. in [3,4], and this line of work was continued by [9,13] who considered additional constraints on the cost functions involved. Gupta and Kumar [10] considered the problem of sorting objects in the case where the comparison costs form a *metric* on the ground set. This is of particular relevance to our work in this paper, where we consider transposition costs which form a metric on the ground set. However, all of the problems in this line of work deal with nonuniform costs on *information*, while we instead impose nonuniform costs on the rearrangement steps, without imposing any costs on information. As such, while the broad ideas are similar, the technical details differ greatly between this line of research and the current work.

A number of practical problems call for positive costs (weights) on transpositions, and costs that capture some constraint on the structure of the transpositions. The problem at hand may then be described as follows: For a given set of positive costs assigned to transpositions of distinct elements, find a smallest cost sequence of transpositions converting a given permutation to the identity.

In our subsequent analysis, we focus on constraints that take into account that elements of the ground set may be similar and that transposing similar elements should induce a smaller cost than transposing dissimilar elements. We refer to the underlying family of distance measures as *similarity distances*. The similarity distance is not to be confused with the distance used in [17], where the goal was to *rank similar items* close to each other in an aggregated list.

The contributions of this work are three-fold. First, we introduce a Y-tree (i.e., a tree with at most one node of degree three) cost function and a notion of similarity between permutations associated with this special tree structure. In this setting, the cost of transposing two elements equals the weight of the shortest path in a Y-tree. Our focus on Y-trees is largely motivated by the fact that the general tree analysis appears to be quite complex. While Y-trees are simple enough to be computationally tractable, they are complex enough that interesting new phenomena arise that are not present in path metrics. Second, we describe an exact linear time decomposition algorithm for cycle permutations with Y-tree costs. Third, we develop a linear time, 4/3-approximation method for computing the similarity distance between arbitrary permutations.

The paper is organized as follows. Section 2 introduces the notation and definitions used throughout the paper. Section 3 contains a brief review of prior work as well as some relevant results used in subsequent derivations. This section also presents a linear time algorithm for computing the Y-tree similarity between cycle permutations. This algorithm is extended in Section 4 to general permutations via cycle-merging strategies that provide linear time, constant-approximation guarantees. Section 5 contains the concluding remarks.

2. Mathematical preliminaries

For a given ground set $[n] \triangleq \{1, 2, \ldots, n\}$, a permutation $\pi : [n] \to [n]$ is a bijection on and onto $[n]$. The collection of all permutations on $[n]$ – the symmetric group of order $n!$ – is denoted by S_n.

There are several ways to represent a permutation. The two-line representation has the domain written out in the first line and the corresponding image in the second line. For example, the following permutation is given in two-line form:

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 1 & 2 & 5 & 4 & 3 \end{pmatrix}.$$

The one-line representation is more succinct as it only utilizes the second row of the two-line representation; the above permutation in one-line format reads as $(6, 1, 2, 5, 4, 3)$. The symbol e is reserved for the identity permutation $(1, 2, \ldots, n)$.

Sometimes, we find it useful to describe a permutation in terms of elements and their images: hence, a third description of the aforementioned permutation is $\pi(1) = 6,\ \pi(2) = 1,\ \pi(3) = 2,\ \pi(4) = 5,\ \pi(5) = 4$, and $\pi(6) = 3$. A straightforward interpretation of these expressions is that $\pi(i)$ represents the element placed in position i. We also define the inverse of a permutation π, π^{-1}, in which $\pi^{-1}(i)$ describes the position of element i. With this notation at hand, the product of two permutations $\pi, \sigma \in S_n$, $\mu = \pi\,\sigma$, can be defined by $\mu(i) = \pi(\sigma(i))$, for all $i \in [n]$. The *support* of a permutation $\pi \in S_n$, written $\mathrm{supp}(\pi)$, is the set of all $i \in [n]$ with $\pi(i) \neq i$. We write $|\pi|$ to refer to $|\mathrm{supp}(\pi)|$.

For $k > 1$, a *k-cycle*, denoted by $\kappa = (i_1\ \ldots\ i_k)$, is a permutation that acts on $[n]$ in the following way[2] :

$$i_1 \to i_2 \to \ldots \to i_k \to i_1,$$

where $x \to y$ denotes $y = \kappa(x)$. In other words, $\kappa = (i_1\ \ldots\ i_k)$ cyclically shifts elements in the permutation confined to the set $\{i_1, \ldots, i_k\}$ and keeps all other elements fixed. A cycle of length 2 is called a *transposition*, and is denoted by $(a\,b)$.

In general, for $a, b \in [n]$, $\pi(a\,b) \neq (a\,b)\pi$, because $\pi(a\,b)$ corresponds to swapping elements of π in positions a and b while $(a\,b)\pi$ corresponds to swapping elements a and b in π. For instance, $(6, 1, 2, 5, 4, 3)(2\,3) = (6, 2, 1, 5, 4, 3)$, while $(2\,3)(6, 1, 2, 5, 4, 3) = (6, 1, 3, 5, 4, 2)$. Note that in the former example, we used $\pi(a\,b)$ to denote the product of a permutation and a transposition.

Two cycles are said to be *disjoint* if the intersection of their supports is empty; furthermore, two cycles are termed to be *adjacent* if they have exactly one common element in their supports. Although non-disjoint cycles are sporadically mentioned in the combinatorial literature, their use is extremely limited due to the fact that disjoint cycles offer simpler

[2] This is not to be confused with the one line representation using commas between entries.