Decomposing Permutations via Cost-Constrained Transpositions

Farzad Farnoud (Hassanzadeh)
Department of Electrical and
Computer Engineering
University of Illinois at Urbana-Champaign
Email: hassanz1@uiuc.edu

Olgica Milenkovic
Department of Electrical and
Computer Engineering
University of Illinois at Urbana-Champaign
Email: milenkov@uiuc.edu

***Abstract*—We consider the problem of finding the minimum cost transposition decomposition of a permutation. In this framework, arbitrary non-negative costs are assigned to individual transpositions and the task at hand is to devise polynomial-time, constant-approximation decomposition algorithms. We describe a polynomial-time algorithm based on specialized search strategies that constructs the optimal decomposition of individual transpositions. The analysis of the optimality of decompositions of single transpositions uses graphical models and Menger's theorem. We also present a dynamic programing algorithms that finds the minimum cost, minimum length decomposition of a cycle and show that this decomposition represents a 4-approximation of the optimal solution. The results presented for individual cycles extend to general permutations.**

I. Introduction

Permutations are arrangements or orderings of elements, frequently used to describe processes and phenomena in mathematics, computer science, communication theory and bioinformatics [1]–[4].

In order to generate an arbitrary permutation, it suffices to apply a sequence of transpositions - swaps of two elements - to the identity permutation. The sequence of swaps can be reversed in order to recover the identity permutation from the starting permutation. This process is referred to as sorting by transpositions, while the inverse process is known as decomposition.

It is straightforward to show that the number of transpositions needed to sort a permutation is the difference of the size of the permutation and the number of cycles formed by the elements of the permutation. We are interested in a related, yet substantially more difficult question: assuming that each transposition has an arbitrary non-negative cost, find the minimum sorting cost and the sequence of transpositions used in this sorting. To the best of the authors' knowledge, the complexity of this problem is not known. In a companion paper [5] we showed that large families of cost functions – such as costs based on metric-paths – have polynomial-time *exact* decomposition algorithms. In this work, we devise algorithms for approximating the minimum decomposition cost for arbitrary non-negative cost functions. The approximation constant does not exceed four.

Our investigation is motivated by three different applications. The first application pertains to sorting of genomic sequences, while the second application is related to a generalization of the notion of a trapdoor channel [6]. The third application is in the area of coding for storage devices [7].

The paper is organized as follows: Section II introduces the relevant definitions and terminology. Section III is devoted to the study of optimal decomposition algorithms for individual transpositions, while Section IV describes constant approximation algorithm for optimizing the cost of a cycle. In Section V, we present a decomposition algorithm for arbitrary permutations. In the discussions to follow, some proofs are omitted for brevity but are presented in the longer arXiv version of this paper [8].

II. Notation and Definitions

A permutation π of $[n] := \{1, 2, \cdots, n\}$ is a bijection from $[n]$ to itself. The *functional digraph* of a function $f : [n] \to [n]$, denoted by $\mathscr{G}(f)$, is a directed graph with vertex set $[n]$ and an edge from i to $f(i)$ for each $i \in [n]$. For a permutation π of $[n]$, $\mathscr{G}(\pi)$ is a collection of disjoint *cycles* since the in-degree and out-degree of each vertex is exactly one. The cycles of a permutation are the cycles of its functional digraph. Often, we do not make a distinction between a permutation with at most one cycle with length larger than one and its longest cycle. In this sense, the composition of individual cycles is well defined. Each cycle σ can be written as a $k-$tuple $(a_1 a_2 \cdots a_k)$, where k is the length of the cycle and $a_{i+1} = \sigma(a_i)$. The cycle representation of a permutation is the list of its cycles (Figure 1a). For each cycle of length k, the indices are evaluated modulo k, so that a_{k+1} equals a_1.

A cycle of length two is called a *transposition*. A *transposition decomposition* (or simply a decomposition) τ of length m of a permutation π is a sequence $t_m \cdots t_1$ of transpositions t_i whose product (i.e., composition, from right to left) is π. Alternatively, a *sorting* s of a permutation π is a sequence of transpositions that transform π into $\imath$, where $\imath$ denotes the identity permutation. In other words, $s\pi = \imath$. Note that a decomposition in reverse order equals a sorting.

An *embedding* is a drawing of a graph in which no two edges cross. An embedding of $\mathscr{G}(\pi)$ can be obtained by placing vertices of each of the disjoint cycles on disjoint circles. We use $\mathscr{G}(\pi)$ to refer to the embedding of the

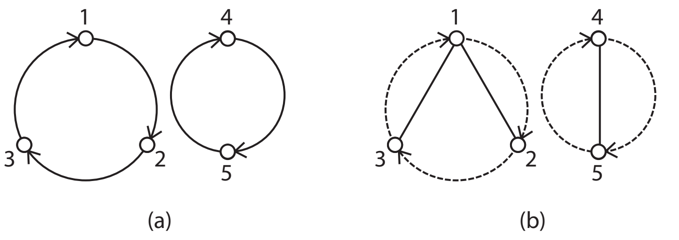


Figure 1: (a) The digraph $\mathscr{G}(\pi)$ of the permutation $\pi = (123)(45)$. (b) The graph $\mathscr{T}(\tau)$, for the decomposition $\tau = (13)(12)(45)$ of π. Edges of $\mathscr{G}(\pi)$ and $\mathscr{T}(\tau)$ are represented with dashed and solid lines, respectively.

functional digraph of π on circles, as well as the functional digraph of π.

Let $\mathscr{T}(\tau)$ be a graph with vertex set $[n]$ and edges $(a_i b_i)$ for each transposition $t_i = (a_i b_i)$ of τ. The drawing of $\mathscr{T}(\tau)$ with the same vertex set as $\mathscr{G}(\pi)$ is also denoted by $\mathscr{T}(\tau)$ (Figure 1b). In the derivations to follow, we use the words transposition and edge interchangeably.

We are concerned with the following problem. One is given a non-negative cost function φ on the set of transpositions. The cost of a decomposition is defined as the sum of the costs of its transpositions. The task is to find an efficient algorithm for generating the Minimum Cost Transposition Decomposition (MCD) of a permutation π. The cost of the MCD of a permutation π under the cost function φ is denoted by $M_\varphi(\pi)$. If there is no ambiguity, the subscript is omitted.

For a non-negative cost function φ, let $\mathscr{K}(\varphi)$ be the undirected complete graph with vertex set $[n]$ in which the cost of each edge (ab) equals $\varphi(a,b)$. The cost of a graph $G \subseteq \mathscr{K}(\varphi)$ is the sum of the costs of its edges,

$$\operatorname{cost}(G) = \sum_{(ab)\in G} \varphi(a,b).$$

The shortest path, i.e., the path with minimum cost, between i and j in $\mathscr{K}(\varphi)$ is denoted by $p^*(i,j)$.

The predecessor of a in $\mathscr{G}(\pi)$ is the unique element x such that $(xa) \in \mathscr{G}(\pi)$, i.e., $\pi(x) = a$. Applying a transposition (ab) to a permutation π is equivalent to exchanging the predecessors of a and b in $\mathscr{G}(\pi)$. Consider the permutation π, where $\pi(c) = a$ and $\pi(d) = b$, for $a, b, c, d \in [n]$. Let $\pi' = (ab)\pi$. In π', the predecessor of a and b are exchanged, so $\pi'(d) = a$ and $\pi'(c) = b$. We define a generalization of the notion of a transposition, termed *h-transposition*, where the predecessor of a can be changed independently of the predecessor of b. Let $\pi'' = (c,(a \to b))\pi$ where $(c,(a \to b))$ represents an h-transposition. The h-transposition takes c, the predecessor of a, to b without modifying the predecessor of b. That is, in π'' we have $\pi''(c) = \pi''(d) = b$, i.e., b has two predecessors, and a has no predecessor. Note that π' is no longer a bijection, but a mapping in which several elements may be mapped to one element. A transposition represents a pair of h-transpositions. For example,

$$(ab)\pi = \left(\pi^{-1}(a), (a \to b)\right)\left(\pi^{-1}(b), (b \to a)\right)\pi.$$

An *h-decomposition* h of a permutation π is a sequence of h-transpositions such that $h\imath = \pi$. The cost assigned to an h-transposition $(c,(a \to b))$, where c is a predecessor of a, is one half of the cost of the corresponding transposition (a,b). Note that the cost of $(c,(a \to b))$ does not depend on c.

For a permutation π and a transposition (ab), it can be easily verified that $(ab)\pi$ consist of one more (or one less) cycle than π if and only if a and b are in the same cycle (in different cycles). Since the identity permutation has n cycles, a *Minimum Length Transposition Decomposition (MLD)* of π has length $n-\ell$, where ℓ denotes the number of cycles in π. The minimum cost of an MLD of π, with respect to cost function φ, is denoted by $L_\varphi(\pi)$. For example, $(132)(45) = (45)(23)(12)$ is decomposed into three transpositions. In particular, if π is a single cycle, then the MLD of the cycle has length $n-1$. A given cycle of length k has k^{k-2} MLDs [9]. An MCD is not necessarily an MLD, as illustrated by the following example.

Example 1. Consider the cycle $\sigma = (1 \cdots 5)$ with $\varphi(i,j) = 3$, for $|i-j| = 1$, and $\varphi(i,j) = 1$ otherwise. It is easy to verify that the decomposition $(14)(13)(35)(24)(14)(13)$ is an MCD of σ with cost six, i.e., $M(\sigma) = 6$. However, as we shall see later, the cost of a minimum cost MLD is eight, i.e., $L(\sigma) = 8$. One such MLD is $(14)(23)(13)(45)$ [10].

III. Optimizing the Cost of Individual Transpositions

Let (ab) be a transposition in a decomposition τ. It can be shown that (ab) has no decompositions of length two and the only decomposition of length one is (ab) itself. It is, then, straightforward to see that the shortest non-trivial decompositions of (ab) must be of the form

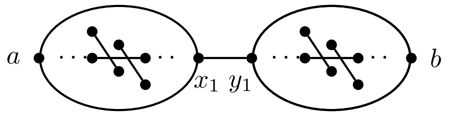
$$(ab) = (ac)(bc)(ac), \tag{1}$$

where $c \in [n]$ and $c \neq a, b$, up to reversing the roles of a and b.

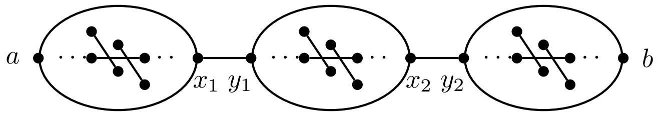
If $\varphi(ab) > 2\varphi(ac) + \varphi(bc)$, then replacing (ab) in τ by $(ac)(bc)(ac)$ reduces the cost associated with τ. Thus, the first step of our decomposition algorithm is to find the optimal cost of each transposition. It is straightforward to develop an algorithm for finding *minimum cost decompositions of transpositions that can be obtained by successive substitutions of form* (1). One such algorithm – Alg. 1 – performs a simple search on the ordered set of transpositions in order to check if their product, of the form (1), yields a decomposition of lower cost for some transposition. It then updates the costs of transpositions and performs a new search for decompositions of length three that may reduce the cost of some transposition.

The input to Alg. 1 is an ordered list Ω of transpositions and their costs. Each row of Ω corresponds to one transposition and is of the form $[(ab)|\varphi(a,b)]$. Sorting of Ω means reordering its rows so that transpositions are sorted in increasing order of their costs. The output of the algorithm is a list with the same format, but with minimized costs for each transposition.

Lemma 2. *Alg. 1 optimizes the costs φ of all transpositions with respect to the triple transposition decomposition* (1).



(a) The edge (x_1y_1) is the only cut-edge



(b) Edges (x_1y_1) and (x_2y_2) are the cut-edges

Figure 2: Illustration for the proof of Theorem 3.

Algorithm 1 OPTIMIZE-TRANSPOSITION-COSTS(Ω)

```
1:  Input: $\Omega$ (the list of transpositions and their cost)
2:  Sort $\Omega$
3:  for $i \leftarrow 2 : |\Omega|$ do
4:      $(a_1b_1) \leftarrow \Omega(i)$
5:      $\phi_1 \leftarrow \varphi(a_1, b_1)$
6:      for $j = 1 : i-1$ do
7:          $(a_2b_2) \leftarrow \Omega(j)$
8:          $\phi_2 \leftarrow \varphi(a_2, b_2)$
9:          if $\{a_1b_1\} \cap \{a_2b_2\} \neq \emptyset$ then
10:             $a_{com} \leftarrow \{a_1, b_1\} \cap \{a_2, b_2\}$
11:             $\{a_3, b_3\} \leftarrow \{a_1, a_2, b_1, b_2\} - \{a_{com}\}$
12:             if $\phi_1 + 2\phi_2 < \varphi(a_3, b_3)$ then
13:                 update $\varphi(a_3, b_3)$ in $\Omega$
14:     Sort $\Omega$
```

Since transposition costs are arbitrary non-negative values, it is not clear that the minimum cost decomposition of a transposition is necessarily generated by Alg. 1. This algorithm only guarantees that one can identify the *optimal sequence of consecutive replacements of transpositions by triples of transpositions*. Hence, the minimum cost of a transposition (ab) may be smaller than $\varphi^*(a, b)$, the cost obtained via Alg. 1. Fortunately, this is not the case, as shown in the following theorem.

Theorem 3. *Alg. 1 produces the minimum cost decompositions of all transpositions.*

Proof: Let $\mathscr{M}$ represent the multigraph of a decomposition for a transposition (ab), having vertices in $[n]$ and edges corresponding to the transpositions used in the decomposition. We will show that the multigraph $\mathscr{M}$ cannot have more than one a,b-cut edge, i.e., one edge whose removal disconnects the vertices a and b. If $\mathscr{M}$ has no a,b-cut edge, then there exist at least two edge-disjoint paths between a and b in $\mathscr{M}$. This claim follows by invoking Menger's theorem which states that the minimum number of edges one needs to delete from a graph G to disconnect two given vertices is also the maximum number of pairwise edge-disjoint paths between those vertices. This theorem holds for multigraphs as well [11]. As shown below, the costs of the paths can be combined, leading to a cost of the form induced by a transposition decomposition optimized via (1). The case when the multigraph has exactly one a,b-cut edge, can be reduced to the case of no a,b-cut edge.

First, we explain how an a,b-cut edge imposes a certain structure on the decomposition of (ab). It can be easily shown that in the multigraph of any decomposition of (ab), there exists a path between a and b. Consider the decomposition $t_mt_{m-1}\cdots t_i\cdots t_1$ of (ab) and suppose that $t_i = (x_1y_1)$ is an a,b-cut edge as shown in Figure 2a. For $1 \leq j \leq m$, let $\pi_j = t_j\cdots t_1$. Since there exists a path between a and b, there also exists a path between a and x_1 that does not use the edge (x_1y_1). Thus, in $\mathscr{M} - (x_1y_1)$, a and x_1 are in the same "component". Denote this component by B_1. Similarly, a component, denoted by B_2, must contain both b and y_1. Since there is no transposition in π_{i-1} with endpoints in both B_1 and B_2, there is no element $z \in B_1$ such that $\pi_{i-1}(z) \in B_2$. Similarly, there is no element $z \in B_2$ such that $\pi_{i-1}(z) \in B_1$. This implies that $\pi_{i-1}(a) \in B_1$ and $\pi_{i-1}(b) \in B_2$. Since (x_1y_1) is the only edge connecting B_1 and B_2, we must have

$$\pi_{i-1}(a) = x_1, \qquad \pi_{i-1}(b) = y_1,$$
$$\pi_i(a) = y_1, \qquad \pi_i(b) = x_1.$$

Also, for all $j \geq i$, we must have $\pi_j(b) \in B_1$.

Now suppose there are at least two a,b-cut edges in $\mathscr{M}$ as shown in Figure 2b. Let the decomposition of (ab) be $t_m\cdots t_l\cdots t_i\cdots t_1$, where $t_i = (x_1y_1)$ and $t_l = (x_2y_2)$ are a,b-cut edges, for some $i < l$. Define B_1, B_2, and B_3 to be the components containing a, y_1, and y_2, respectively, in $\mathscr{M} - (x_1y_1) - (x_2y_2)$. By the same reasoning as above we must have

$$\pi_{l-1}(a) = x_2, \qquad \pi_{l-1}(b) = y_2.$$

However, this cannot be true since for all $j \geq i$, we have $\pi_j(b) \in B_1$ and thus $\pi_{l-1}(b) \neq y_2 \in B_3$. This contradiction shows that $\mathscr{M}$ cannot contain more than one a,b-cut edge.

Consider the case when there is no a,b-cut edge in $\mathscr{M}$. In this case, based on Menger's theorem, there must exist at least two pairwise edge disjoint paths between a and b. The cost of one of these paths has to be less than or equal to the cost of the other. Refer to this path as the *minimum path*. Clearly the cost of the decomposition of (ab) described by $\mathscr{M}$ is greater than or equal to twice the cost of the minimum path. Let the edges of the minimum path be $(az_1)(z_1z_2)...(z_{m-1}z_m)(z_mb)$ for some integer m. The cost of (ab) is greater than or equal to

$$\begin{aligned} 2\varphi^*(a, z_1) + 2\varphi^*(z_1, z_2) + \cdots + 2\varphi^*(z_{m-1}z_m) + 2\varphi^*(z_m, b) \geq \\ \varphi^*(a, z_1) + 2\varphi^*(z_1, z_2) + ... + 2\varphi^*(z_{m-1}z_m) + 2\varphi^*(z_m, b) \geq \\ \varphi^*(a, z_2) + 2\varphi^*(z_2, z_3) + ... + 2\varphi^*(z_m, b) \geq \\ \cdots \geq \varphi^*(a, z_m) + 2\varphi^*(z_m, b) \geq \varphi^*(a, b), \end{aligned}$$

and the cost of the decomposition associated with $\mathscr{M}$ cannot be smaller than the cost of the optimal decomposition produced by Alg. 1. Next, consider the case when there is one a,b-cut edge in $\mathscr{M}$. In this case, we distinguish two scenarios: when $x_1 = a$, and when $x_1 \neq a$. In the former case, the transposition (ay_1) plays the role of the transposition (az_1) and the remaining transpositions used in the decomposition lie in the graph $\mathscr{M} - (ay_1)$. Since $\mathscr{M} - (ay_1)$ has no a,b-cut edge, continuing with line two of (6) proves that the cost of the decomposition associated with $\mathscr{M}$ cannot be smaller than $\varphi^*(a,b)$. The latter case can be handled similarly. ■

It can be shown that the complexity of Alg. 1 equals $O(n^4)$. We find the following lemma useful in Section IV.

Lemma 4. *The minimum cost of a transposition is at most twice the cost of the shortest path between its elements in $\mathscr{K}(\varphi)$. That is, $\varphi^*(a,b) \leq 2\,\mathrm{cost}\,(p^*(a,b))$.*

Proof: Suppose that $p^*(a,b) = c_0 c_1 \cdots c_m c_{m+1}$ where $a = c_0$ and $b = c_{m+1}$. It is easy to see that

$$(ab) = (c_0c_1)(c_1c_2)\cdots(c_mc_{m+1})(c_mc_{m-1})\cdots(c_1c_0)\cdot \quad (2)$$

So the right-hand side of (2) is a decomposition of (ab). The cost of this decomposition is

$$2\sum_{j=0}^{m}\varphi(c_j,c_{j+1}) - \varphi(c_m,c_{m+1}) \leq 2\sum_{j=0}^{m}\varphi(c_j,c_{j+1})$$

and since the right-hand side is equal to $2\,\mathrm{cost}\,(p^*(a,b))$, the proof is complete. ■

IV. Optimizing the Cost of Cycles

We consider next the cost optimization problem over single cycles. First, we find the minimum cost MLD via a dynamic programming algorithm. The minimum cost MLD is obtained with respect to the optimized cost function φ^* of the previous section.

The results in this section apply to any cycle σ. However for clarity of presentation, and without loss of generality, we consider the cycle $\sigma = (12\cdots k)$.

A. Minimum Cost MLDs

Recall that the vertices of $\mathscr{G}(\sigma)$ are placed on a circle.

Lemma 5. *For an MLD $\tau = t_1 \cdots t_{k-1}$ of σ, $\mathscr{T}(\tau)$ is a tree. Furthermore, $\mathscr{T}(\tau) \cup \mathscr{G}(\sigma)$ is planar. That is, for $t_i = (a_1a_2)$, where $a_1 < a_2$ and $t_j = (b_1b_2)$, where $b_1 < b_2$, if $a_1 < b_1 < a_2$ then $a_1 < b_2 < a_2$.*

The following lemma, proved in our companion paper [5], establishes a partial converse to the previous lemma.

Lemma 6. *For a cycle σ and a spanning tree T over the vertices $\{1,2,\cdots,k\}$, for which $\mathscr{G}(\sigma) \cup T$ is planar, there exists at least one MLD τ of σ such that $T = \mathscr{T}(\tau)$.*

For related ideas regarding permutation decompositions and graphical structures, the interested reader is referred to [12].

Since any MLD of a cycle can be represented by a tree that is planar on the circle, the search for an MLD of minimum cost only needs to be performed over the set of planar trees. This search can be performed using a dynamic program, outlined in Alg. 2. Lemma 7 establishes that Alg. 2 produces a minimum cost MLD.

Algorithm 2 MIN-COST-MLD

```
1:  Input: Optimized transposition cost function $\Phi^*$, where
    $\Phi^*_{i,j} = \varphi^*(i,j)$ (Output of Alg. 1)
2:  $C(i,j) \leftarrow \infty$ for $i,j \in [k]$
3:  $C(i,i) \leftarrow 0$ for $i \in [k]$
4:  $C(i,i+1) \leftarrow \varphi^*(i,i+1)$ for $i \in [k]$
5:  for $l = 2\cdots k-1$ do
6:      for $i = 1\cdots k-l$ do
7:          $j \leftarrow i+l$
8:          for $i \leq s < r \leq j$ do
9:              $A \leftarrow C(i,s)+C(s+1,r)+C(r,j)+\varphi^*(i,r)$
10:             if $A < C(i,j)$ then
11:                 $C(i,j) \leftarrow A$
```

Lemma 7. *The output cost of Alg.* (2) $C(1,k)$, *equals* $L(\sigma)$.

Proof: The algorithm finds the minimum cost MLD of $(1\cdots k)$ by first finding the minimum cost of MLDs of shorter cycles of the form $(i\cdots j)$, where $1 \leq i < j \leq k$. We look at the computations performed in the algorithm from a top-down point of view.

Let $C_T(i,j)$ be the cost of the decomposition of the cycle $\sigma^{i,j} = (i\cdots j)$, using edges of $T[\{i,\cdots,j\}]$ where T is an arbitrary planar spanning tree over the vertices $\{1,\cdots,k\}$ arranged on a circle. For a fixed T, let

$$r = \max\{u|\,(1u) \in T\}.$$

Since T is a tree, $T - (1r)$ has two components. These two components have vertex sets $\{1,\cdots,s\}$ and $\{s+1,\cdots,k\}$, for some s. It is easy to see that

$$(1\cdots k) = (s+1\cdots k\,1)(1\cdots s). \quad (3)$$

We may write

$$(i\cdots j) = (s+1\cdots r)(ir)(r\cdots j)(i\cdots s) \quad (4)$$

where $i \leq s < r \leq j$. Thus

$$C_T(i,j) = C_T(s+1,r) + \varphi^*(i,r) + C_T(r,j) + C_T(i,s).$$

Define $C(i,j) = C_{T^*}(i,j)$, where $T^* = \arg\min_T C_T(i,j)$ denotes a tree that minimizes the cost of the decomposition of $(i\cdots j)$. Then, we have

$$C(i,j) = C(s^*+1,r^*) + \varphi^*(i,r^*) + C(r^*,j) + C(i,s^*), \quad (5)$$

where s^* and r^* are the values that minimize $C_T(i,j)$ under the constraint $1 \leq i \leq s < r \leq j$. Since the cost of each cycle can be computed from the cost of shorter cycles $C(i,j)$ can be obtained recursively, with initialization

$$C(i,i+1) = \varphi^*(i,i+1). \quad (6)$$

The algorithm searches over s and r and computes $C(1,k)$ using (5) and (6). Although these formulas are written in a recursive form, Alg. 2 is written as a dynamic program. The algorithm first computes $C(i,j)$ for small values of i and j, and then finds the cost of longer cycles. That is, for each $2 \leq l \leq k-1$ in increasing order, $C(i,i+l)$ is computed by choosing its optimal decomposition in terms of costs of smaller cycles. ■

It can be shown that the complexity of Alg. 2 is $O\left(k^4\right)$.

B. Constant-factor approximation for cost of MCD

For the cycle $\sigma = (12\cdots k)$ and $1 \leq j \leq k$ consider the decomposition

$$\begin{aligned}(j+1\ j+2)(j+2\ j+3)\cdots\\ (k-1\ k)(k1)(12)(23)\cdots(j-1\ j). \quad (7)\end{aligned}$$

The cost of this decomposition equals $\sum_{i\in\sigma}\varphi^*(i,\sigma(i)) - \varphi^*(j,\sigma(j))$.

To minimize the cost of the decomposition, we choose j such that the transpositions $(j\ j+1)$ has maximum cost. That is, we let $j = j^*$ in (7) where $j^* = \arg\max_{j\in\sigma}\varphi^*(j,\sigma(j))$. This decomposition is termed the Simple Transposition Decomposition (STD) of σ and its cost is denoted by $S(\sigma)$.

Theorem 8. *For a cycle σ $M(\sigma) \leq L(\sigma) \leq S(\sigma) \leq 4M(\sigma)$.*

Proof: Clearly, $M(\sigma) \leq L(\sigma)$. It is easy to see that the STD is itself an MLD and, thus, $L(\sigma) \leq S(\sigma)$. For $S(\sigma)$, we have

$$\begin{aligned}S(\sigma) &= \sum_{i\in\sigma}\varphi^*(i,\sigma(i)) - \varphi^*(j^*,\sigma(j^*))\\ &\leq \sum_{i\in\sigma}\varphi^*(i,\sigma(i)) \leq 2\sum_{i\in\sigma}\mathrm{cost}(p^*(i,\sigma(i)))\end{aligned}$$

where the last inequality follows from Lemma 4. To complete the proof, we need to show that $M(\sigma) \geq \frac{1}{2}\sum_i \mathrm{cost}(p^*(i,\sigma(i)))$. Since this result is of independent importance we state it in Lemma 9. ■

Recall from Section II that

$$\left(\sigma^{-1}(a),(a\to b)\right)\left(\sigma^{-1}(b),(b\to a)\right)\sigma = (ab)\sigma$$

and that the cost of each h-transposition is half the cost of the corresponding transposition.

Lemma 9. *It holds that $M(\sigma) \geq \frac{1}{2}\sum_i \mathrm{cost}(p^*(i,\sigma(i)))$.*

Proof: Any decomposition can be written as an h-decomposition with the same cost by breaking each transposition into two h-transpositions. Thus, the minimum cost of a decomposition, $M(\sigma)$, is at least as large as the minimum cost of an h-decomposition. The minimum cost h-decomposition uses the shortest path $p^*(i,\sigma(i))$ between i and $\sigma(i)$. In this case, i becomes the predecessor of $\sigma(i)$ through the following sequence of h-transpositions:

$$(i,(v_m\to\sigma(i)))\cdots(i,(v_1\to v_2))(i,(i\to v_1)),$$

where $p^*(i,\sigma(i)) = iv_1v_2\cdots v_m\sigma(i)$ is the shortest path between i and $\sigma(i)$. This h-decomposition has cost

$$\frac{1}{2}\sum_i \mathrm{cost}(p^*(i,\sigma(i))),$$

which completes the proof. ■

V. Optimizing the Cost of Permutations

Most of the results in the previous section generalize to permutations with multiple cycles without difficulty.

Let π be a permutation, with cycle decomposition $\sigma_1\sigma_2\cdots\sigma_\ell$. A decomposition of π with minimum number of transpositions is the product of MLDs of individual cycles σ_i. Thus, the minimum cost MLD of π equals $L(\pi) = \sum_{t=1}^{\ell} L(\sigma_t)$. Furthermore, the STD of π is the product of the STDs of individual cycles σ_i.

Theorem 10. *Consider a permutation π with cycle decomposition $\sigma_1\sigma_2\cdots\sigma_\ell$ and cost function φ. The following claims hold.*

1) $S(\pi) \leq 2\sum_i \mathrm{cost}(p^*(i,\pi(i)))$.
2) $M(\pi) \geq \frac{1}{2}\sum_i \mathrm{cost}(p^*(i,\pi(i)))$.
3) $L(\pi) \leq S(\pi) \leq 4M(\pi)$.

Theorem 10 indicates that a minimum cost MLD represents a good approximation for an MCD, independent of the choice of the cost function. This holds true *only if* the costs of individual transpositions are first optimized using Alg. 1. Note that in this case, the decomposition of a single transposition itself can be of length greater than or equal to three.

Acknowledgement. The authors gratefully acknowledge discussions with Chandra Chekuri and Navin Kashyap. This work was funded by the NSF grants NSF CCF 08-21910 and NSF CCF 08-09895.

References

[1] I. P. Goulden and D. M. Jackson, *Combinatorial enumeration*. Dover Pubns, 2004.

[2] J. H. van Lint and R. M. Wilson, *A course in combinatorics*. Cambridge Univ Pr, 2001.

[3] F. R. K. Chung, “An algebraic approach to switching networks.”

[4] M. Hofri, *Analysis of algorithms: Computational methods and mathematical tools*. Oxford University Press Oxford, UK, 1995.

[5] F. Farnoud, C.-Y. Chen, O. Milenkovic, and N. Kashyap, “A graphical model for computing the minimum cost transposition distance,” in *Information Theory Workshop (ITW), 2010 IEEE*, 30 2010-sept. 3 2010, pp. 1 –5.

[6] H. Permuter, P. Cuff, B. V. Roy, and T. Weissman, “Capacity of the trapdoor channel with feedback.” *IEEE Transactions on Information Theory*, vol. 54, no. 7, pp. 3150 – 3165, 2008.

[7] A. Barg and A. Mazumdar, “Codes in permutations and error correction for rank modulation,” *Information Theory, IEEE Transactions on*, vol. 56, no. 7, pp. 3158 –3165, july 2010.

[8] F. Farnoud and O. Milenkovic, “Sorting of permutations by cost-constrained transpositions,” *Arxiv preprint arXiv:1007.4236*, 2010.

[9] N. Meier and J. Tappe, “Ein Neuer Beweis der Nakayama-Vermutung Uber die Blockstruktur Symmetrischer Gruppen,” *Bull. London Math. Soc.*, vol. 8, no. 1, pp. 34–37, 1976. [Online]. Available: http://blms.oxfordjournals.org

[10] N. Kashyap, *Personal Communication*, 2010.

[11] D. B. West, *Combinatorial Mathematics*. Preliminary version, 2009.

[12] I. Goulden, “Tree-like properties of cycle factorizations,” *Journal of Combinatorial Theory, Series A*, vol. 98, no. 1, pp. 106 – 117, Apr. 2002.