MATH 674: Final Project
Deletion-Resilient Permutations

Jau Tung Chan, Hannah Lawrence, Julia Wei

May 8, 2019

Abstract

In this paper, we investigate a permutation-based approach to a narrow class of error-correcting codes, proposed by Alon et al. (2018). The premise of these error-correcting codes is the transmission of a message $M = (s_1, s_2, \ldots, s_n)$, a sequence of *distinct* symbols. Using a prior agreed-upon permutation $\sigma : [n] \to [n]$, the sender will send the message M along with the permuted message $\sigma(M) = (s_{\sigma(1)}, s_{\sigma(2)}, \ldots, s_{\sigma(n)})$ through a deletion channel. The channel deletes at most d symbols from each message, producing M' and $\sigma(M)'$ respectively. Alon et al. investigates the necessary and sufficient conditions for the original message M to be recovered, given only σ, M', and $\sigma(M)'$. In our project, we elucidate some of their results more intuitively, extend their results to some more general cases, and finally compare the practical relevance of this type of error-correcting codes to other conventional error-correcting codes.

1 Introduction

In this project, we will investigate a permutation-based approach to a narrow class of error-correcting codes, as proposed in the 2018 paper *Permutations Resilient to Deletions* by Noga Alon, Steve Butler, Ron Graham, and Utkrisht C. Rajkumar [1]. We will first summarize and elucidate some of the results proved in the paper more intuitively, and then consider some extensions of their results to slightly more general cases (with some questions left open). Finally, we will compare the practical significance and relevance of these error-correcting codes in relation to other conventional methods of error-correcting codes.

Broadly speaking, channel coding theory deals with a simple problem of communication. Suppose you wish to communicate some message to a friend, but the only means of doing so is via a noisy channel. How can you add redundancy to your message to alleviate the deleterious effects of the channel, and how efficiently (in number of communication bits required per original message bit) can this be done? Error-correcting codes allow the recipient to reconstruct the original message, even after passing through the imperfect channel.

Error-correcting codes are important in many real-world applications. In everyday life, bar codes that are used on nearly all mass-produced consumer goods use some form of error correcting codes (typically either UPC-A or EAN-13 codes [2]). The same goes for credit card numbers [2], as well as the popular QR Code (at the highest level, up to 30% of the QR code can be damaged and

still recoverable [3]). Beyond these applications we observe in everyday life, satellite broadcasting and deep-space telecommunications also rely heavily on error-correcting codes in order to ensure that the messages sent are correctly received (especially since there can be high error rates when transmitting these messages over long distances) [4].

One conventional method to implement error-correcting codes relies on restricting the domain of possible messages that can be sent from the sender to the receiver. By ensuring that all messages in the domain of possible messages are pairwise sufficiently distinct (which can be quantified in different ways), we can ensure that any noise or error induced by the channel will not be sufficient to make two distinct messages look the same to the receiver (and therefore the receiver will be able to distinguish between distinct messages, based on what they receive). For example, Sloane (2000) investigates binary single-deletion-correcting codes [5]. Similarly, Beame et al. (2009) investigates the maximum value m for which every set of k permutations on $[n]$ is guaranteed to contain a common subsequence of length at most m [6]. k is effectively the maximum size of the domain of possible permutations that can be sent through a deletion channel, so that any deletion of up to $n - m$ symbols can be distinguished pairwise.

On the other hand, deletion-resilient permutations are a specific class of error-correcting codes proposed in [1]. In their formulation, the channel exclusively *deletes* message bits; "redundancy" is added to the message by sending, in addition to the original message, a predetermined permutation of its bits (or in this case, characters).

To be concrete, the message is defined to be $M = (s_1, s_2, \ldots, s_n)$ a sequence of *distinct* symbols, together with a prior agreed-upon permutation $\sigma : [n] \to [n]$ (which is a bijection). The sender will send the message M and the permuted message $\sigma(M) = (s_{\sigma(1)}, s_{\sigma(2)}, \ldots, s_{\sigma(n)})$ through the channel. The channel in this case deletes at most d symbols from each message, producing M' and $\sigma(M)'$ respectively. Naturally, it is assumed that the $\leq d$ symbols deleted from M' and the $\leq d$ symbols deleted from $\sigma(M)'$ are disjoint; if a symbol does not occur in either M' or $\sigma(M)'$, we have no way of knowing its value.

The paper investigates the necessary and sufficient conditions for someone to be able to reconstruct M given σ, M', and $\sigma(M)'$. If this is possible for any M, we say σ is d-resilient.

2 Article Summary

There are two main parts of the article by Alon et al. [1]. They first proved some necessary and sufficient conditions for a permutation to be d-resilient, and then exhibited an algorithm for constructing d-resilient permutations for arbitrarily large d.

2.1 Necessary and sufficient conditions

Suppose that we are given a permutation σ on $[n]$. We would like to determine whether σ is d-resilient. We will often refer to the two-line representation of σ (or just the "two-line representation"), which is given by $\begin{pmatrix} 1 & 2 & \ldots & n \\ \sigma(1) & \sigma(2) & \ldots & \sigma(n) \end{pmatrix}$.

Following Alon et al.'s method, we first define the following bipartite graph $G(\sigma, D)$, for some permutation σ on $[n]$, and any $D \subset [n]$. Let t_1, t_2, ..., t_i be the maximal subsets of D whose elements appear consecutively in $[n]$, i.e. whose elements appear consecutively in the top line of

the two-line representation. Similarly, let b_1, b_2, ..., b_j be the maximal subsets of D whose elements appear consecutively in $\sigma([n])$, or the bottom line of the two-line representation. Then we define

$$V(G(\sigma, D)) := \{t_1, t_2, ..., t_i, b_1, b_2, ..., b_j\}.$$

For all k and l, we then add edge (t_k, b_l) to $G(\sigma, D)$ $|t_k \cap b_l|$ times. Since every element of D appears in exactly one intersection, $G(\sigma, D)$ will have exactly $|D|$ edges.

For example, consider a σ that permutes [9], given by $\sigma = \binom{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9}{7\ 4\ 1\ 8\ 5\ 2\ 9\ 6\ 3}$. Let $D = \{3, 4, 6, 7\}$. Then $G(\sigma, D)$ has vertices $t_1 = \{3, 4\}$, $t_2 = \{6, 7\}$, $b_1 = \{7, 4\}$, $b_2 = \{6, 3\}$, so $G(\sigma, D)$ looks like Figure 1 below.

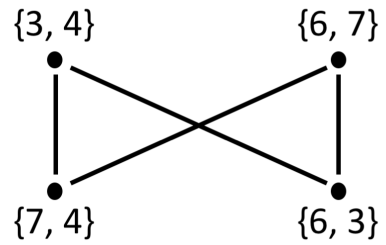

Figure 1: $G(\sigma, D)$, where $\sigma = \binom{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9}{7\ 4\ 1\ 8\ 5\ 2\ 9\ 6\ 3}$ and $D = \{3, 4, 6, 7\}$.

Using this definition of $G(\sigma, d)$, Alon et al. proves the following two theorems.

Theorem 1. *If there exists D such that $|D| \leq 2d$ and $G(\sigma, D)$ has a cycle, then σ is not d-resilient.*

Theorem 2. *If for all $|D| \leq 2d$, the graph $G(\sigma, D)$ is acyclic, then σ is d-resilient.*

Together, these establish necessary and sufficient conditions for σ to be d-resilient, namely, σ is d-resilient iff for all $D \subset [n]$ such that $|D| \leq 2d$, $G(\sigma, D)$ is acyclic. We will now go over the proofs of the theorems in greater detail.

Proof of Theorem 1. Given σ and D such that $|D| \leq 2d$ and $G(\sigma, D)$ has a cycle, Alon et al. provides a method to produce two distinct messages M_1 and M_2 such that $M_1' = M_2'$ and $\sigma(M_1)' = \sigma(M_2)'$, using $\leq d$ deletions for each of the primes. This shows that σ is not d-resilient.

We will walk through the method step by step, illustrated with an example to build more intuition. For ease of notation, we will always pick the message M_1 to be the sequence of symbols (numbers) '$123\ldots n$'. We start with the example depicted in Figure 2.

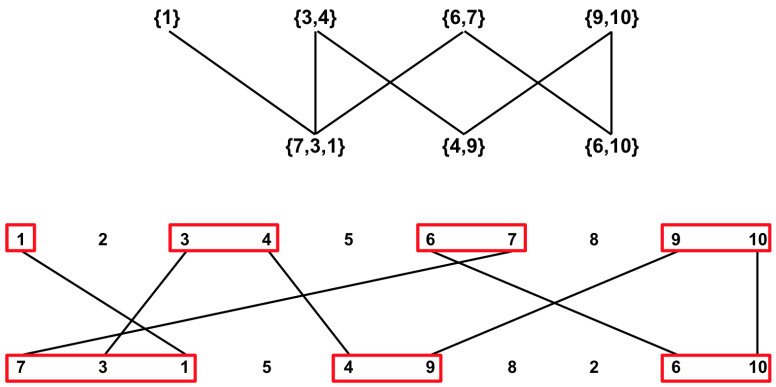

Figure 2: The bottom diagram is the two-line representation of a given $\sigma = \binom{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10}{7\ 3\ 1\ 5\ 4\ 9\ 8\ 2\ 6\ 10}$. We consider some given $D = \{1, 3, 4, 6, 7, 9, 10\}$, with $|D| = 7 \leq 2 \cdot 4$. The top diagram then depicts $G(\sigma, D)$ which clearly contains a 6-cycle. By Theorem 1, we know that having this σ and D implies that σ is not 4-resilient (in fact, it is not even 2-resilient, but this example is just to build intuition). The same edges of $G(\sigma, D)$ are drawn analogously in the bottom diagram, and vertices correspond to the contiguous blocks outlined in red.

We first define an orientation H_1 on $G(\sigma, D)$ (i.e. assign a direction to each edge of $G(\sigma, D)$), using the following two steps (we will use the bottom diagram representation of $G(\sigma, D)$ instead of the top diagram representation, for greater clarity):

- Orient the edges of the cycle in $G(\sigma, D)$ so that each vertex in the cycle has in-degree and out-degree 1.
- Orient the remaining edges, so that there are at most d edges directed into the $\{t_k\}$ collectively, and at most d edges directed into the $\{b_l\}$ collectively. Because the number of edges directed into either the $\{t_k\}$ or the $\{b_l\}$ from the previous step is at most d, while there are at most $2d$ edges total, this can be done.

Our example now looks like Figure 3.

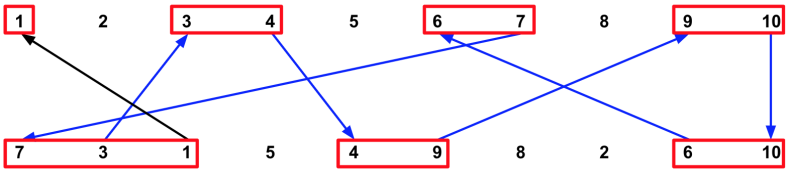

Figure 3: The same $G(\sigma, D)$, now with an orientation H_1 assigned to the edges. The 6-cycle is highlighted in blue.

We carry out the following 4 steps:

1. From this diagram, we let the first row be M_1 and the second row be $\sigma(M_1)$ (consistent with our choice of the message M_1 being '$123\ldots n$'). We can directly obtain M_1' and $\sigma(M_1)'$ by replacing the *tail* of each directed edge with a $*$ (representing a symbol lost in the deletion channel). This looks like Figure 4.

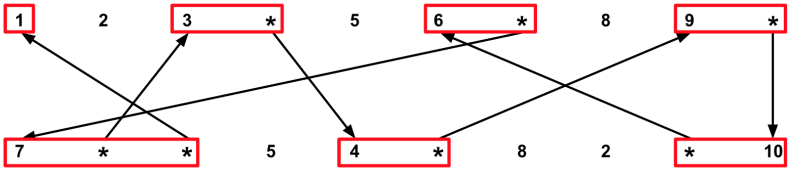

Figure 4: Deriving M_1' and $\sigma(M_1)'$ from $G(\sigma, D)$ as above. In this case, $M_1' = 1\ 2\ 3\ 5\ 6\ 8\ 9$ and $\sigma(M_1)' = 7\ 5\ 4\ 8\ 2\ 10$. Because there are at most d edges directed into the $\{t_k\}$ collectively, and at most d edges directed into the $\{b_l\}$ collectively, there are at most d symbols deleted from either row.

2. Then, from the orientation H_1, we reverse the directions of the edges in the cycle, to get a second orientation H_2. All other directed edges that are not involved in the cycle are left unchanged. An example of this is shown in Figure 5.

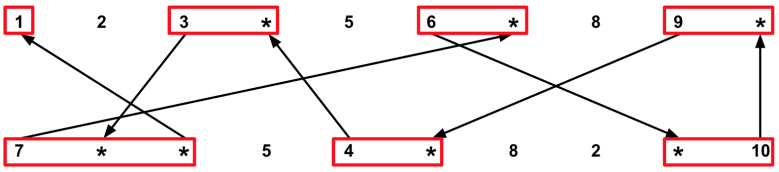

Figure 5: Creating the new orientation H_2 on the same graph $G(\sigma, D)$. Note that reversing the direction of the cycle does not change the in-degree and out-degree of any vertex in this graph.

3. Now, for each vertex t_k and b_l (corresponding to contiguous blocks of symbols), move the non-$*$ symbols, preserving the relative order of those symbols, so they correspond to positions with an edge directed in. Each of the $*$ symbols now has an edge directed out. This is always possible because of the earlier claim that the in-degree and out-degree of every vertex in this graph stays the same. The example looks like Figure 6.

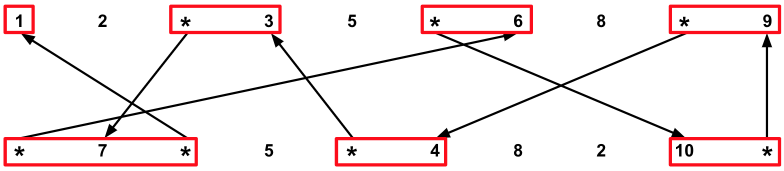

Figure 6: Moving the symbols within each vertex to align the $*$ symbols to edges directed out in H_2.

4. These lines represent M_2' and $\sigma(M_2)'$ respectively, where the $*$ symbols again correspond to symbols that are deleted by the deletion channel. Because we preserve the relative order of the non-$*$ symbols, it is clear that $M_1' = M_2'$ and $\sigma(M_1)' = \sigma(M_2)'$. We can read off M_2 and $\sigma(M_2)$ by noting that $*$'s joined by edges to another (known) symbol is equivalent to (the deleted version of) that symbol. In our example, $M_2 = 1\ 2\ 7\ 3\ 5\ 10\ 6\ 8\ 4\ 9$ and $\sigma(M_2) = 6\ 7\ 1\ 5\ 3\ 4\ 8\ 2\ 10\ 9$. Again, at most d symbols deleted from either row because the in-degree and out-degree of every vertex in this graph stays the same.

Yet $M_1 \neq M_2$ because in Step 3, the location of at least two symbols in M_1 will have changed in M_2, because the orientation of at least two edges will have been changed in Step 2.

We should check one more thing to verify this method: that after Step 4, the bottom row is a permutation of the top row by σ. To check this, we see that the positions of symbols that are not in the cycle of $G(\sigma, D)$ are the same in M_1 and M_2, as well as in $\sigma(M_1)$ and $\sigma(M_2)$. While the positions of symbols that are in the cycle do change, because the assignment of values to $*$ symbols in Step 4 is according to σ, so that the symbol at index i in M_2 is at index $\sigma^{-1}(i)$ in M_2'. So this is true.

We have therefore produced two distinct messages M_1 and M_2 such that $M_1' = M_2'$ and $\sigma(M_1)' = \sigma(M_2)'$, using $\leq d$ deletions for each of the primes, just by utilizing a given $D \subset [n]$ such that $|D| \leq 2d$ and the fact that $G(\sigma, D)$ has a cycle. □

Proof of Theorem 2. Suppose that we have a permutation σ such that, for all $|D| \leq 2d$, the graph $G(\sigma, D)$ is acyclic. We wish to show that σ is d-resilient. To show this, we assume that we are given some M' and $\sigma(M)'$. We will show that the reconstruction of M from these two pieces of information is unique.

First, we show that we can determine the location of all doubly-occurring symbols (i.e. symbols that are present in both M' and $\sigma(M)'$).

Suppose that symbol x occurs in M' at the index of k, where $k \in [n]$. Then, because the deletion channel deletes up to d symbols, we know that the index of x in M is one of $\{k, k+1, k+2, \ldots, k+d\}$.

If the index of x in M is i, then the index of x in $\sigma(M)$, by definition of a permutation of the message, is an index j such that $\sigma(j) = i$. In other words, if the index of x in M is i, then the index of x in $\sigma(M)$ is $\sigma^{-1}(i)$.

Since the index of x in M is one of $\{k, k+1, k+2, \ldots, k+d\}$, the index of x in $\sigma(M)$ is one of $\{\sigma^{-1}(k), \sigma^{-1}(k+1), \sigma^{-1}(k+2), \ldots, \sigma^{-1}(k+d)\}$. For ease of notation, define $a_i := \sigma^{-1}(k+i)$, for $0 \leq i \leq d$.

If the index of x in $\sigma(M)$ is a_i for some i, then, because the deletion channel deletes up to d symbols, the index of x in $\sigma(M)'$ is one of $\{a_i, a_i - 1, a_i - 2, \ldots, a_i - d\}$.

In other words, from the fact that x occurs in M' at the index of k, we can deduce that the index of x in $\sigma(M)'$ is in one of the following sets, in the corresponding cases:

$$\begin{aligned}
S_0 &:= \{a_0, a_0 - 1, a_0 - 2, \ldots, a_0 - d\} \quad \text{(if the index of } x \text{ in } M \text{ is } k\text{)} \\
S_1 &:= \{a_1, a_1 - 1, a_1 - 2, \ldots, a_1 - d\} \quad \text{(if the index of } x \text{ in } M \text{ is } k+1\text{)} \\
S_2 &:= \{a_2, a_2 - 1, a_2 - 2, \ldots, a_2 - d\} \quad \text{(if the index of } x \text{ in } M \text{ is } k+2\text{)} \\
&\vdots \\
S_d &:= \{a_d, a_d - 1, a_d - 2, \ldots, a_d - d\} \quad \text{(if the index of } x \text{ in } M \text{ is } k+d\text{)}
\end{aligned}$$

Note also that since x is doubly occurring, we know the actual index of x in $\sigma(M)'$. So, as long as these sets S_i are all pairwise disjoint, we can uniquely identify the only possible case for the index of x in $\sigma(M)'$ to be in that set S_i. This would allow us to determine the index of x in M (and therefore in $\sigma(M)$ since σ is known).

It remains to check that all these sets S_i are pairwise disjoint. Suppose by way of contradiction that this is not true, i.e. S_i and S_j are not pairwise disjoint for some $1 \leq i, j \leq d$. Then, by inspecting the form of S_i and S_j, we know $|a_i - a_j| \leq d$. This means that, in the two-line representation of σ, there is an element $y := k + i$ and an element $z := k + j$, where y and z have distance at most

d apart in the top line, and the indices of y and z in the bottom line are a_i and a_j respectively, which have distance at most d apart in the bottom line.

We can then construct the a set D consisting of y, z, and all the elements between y and z in both the top and bottom lines. This has size $|D| \leq 2d$ because at most $d+1$ elements are selected from each line and y and z are repeated in both selections. If we construct $G(\sigma, D)$ based on this D, note that y and z are both in the same contiguous block in both the top and bottom lines, so this forms a two-cycle in $G(\sigma, D)$ and contradicts the assumption that for all $|D| \leq 2d$, the graph $G(\sigma, D)$ is acyclic.

Therefore, we now know all the locations of all doubly-occurring symbols. As a result, we also know all the locations of all symbols that are involved in deletions (i.e. singly-occurring symbols).

We can construct a diagram to represent this information, analagous to the two-line representation of σ, as seen in Figure 7.

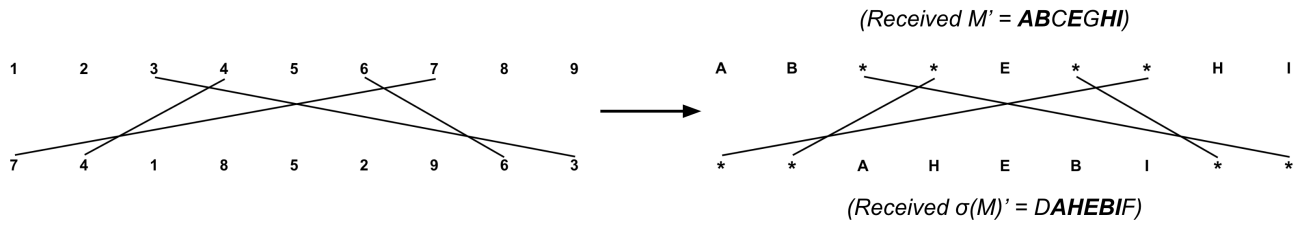

Figure 7: Given the permutation σ (two-line representation on the left), and after plugging in the locations of all doubly-occurring symbols (given by the bolded symbols in M' and $\sigma(M)'$, we denote all other locations as $*$, and join corresponding $*$'s to each other according to the permutation σ.

Let D be the set of singly-occurring symbols in M' and $\sigma(M)'$. By assumption that the deletion channel deleted at most d symbols from each, we know that $|D| \leq 2d$, so by assumption, the graph $G(\sigma, D)$ is acyclic (although this is not the case in Figure 7 above).

We note that the graph $G(\sigma, D)$ corresponds simply to considering each contiguous sequence of $*$'s as one vertex in the bipartite graph, and the edges of $G(\sigma, D)$ are the lines joining corresponding $*$'s as drawn above.

Furthermore, since we receive M' and $\sigma(M)'$, we know how many symbols end up present in each contiguous sequence of $*$'s (i.e. corresponding to the in-degree of that vertex when we orient the edges from deleted symbol to remaining symbol). Together with the total degree of each vertex in the above graph, we know the exact in-degree and out-degree of each vertex.

To show that the reconstruction of M is unique, it suffices to show that there is only one orientation H of the edges in the above graph that satisfies all the (known) in-degree and out-degrees of each vertex. This is because an orientation H of the edges determines the locations of all singly-occurring symbols, and allows us to read off the original message M from the first row of the diagram above.

Alon et al. provides a relatively clear argument for this fact. By way of contradiction, suppose otherwise that there are two distinct orientations H_1 and H_2 of the edges in an acyclic graph, such that for every vertex v in the graph, $in_{H_1}(v) = in_{H_2}(v)$ and $out_{H_1}(v) = out_{H_2}(v)$, where in and

out denote the in-degree and out-degree of a vertex in the respective orientations.

Let e_1 be any edge that has a different direction between H_1 and H_2. Let e_1 be H_1-oriented from vertex v_0 to vertex v_1, and H_2-oriented from v_1 to v_0. Then, considering v_1 has the same in-degree between H_1 and H_2, and the same out-degree between H_1 and H_2, there has to be some other edge e_2 that is H_1-oriented out of v_1 but H_2-oriented into v_1. Let e_2 be H_1-oriented into another vertex v_2, so e_2 is H_2-oriented from v_2 to v_1. This process can be repeated indefinitely to find a sequence $v_0 \to v_1 \to v_2 \to \cdots$ in H_1-orientation. But, because the number of vertices is finite, this will eventually reach a vertex that has previously been seen. This forms a directed cycle in H_1-orientation, but more importantly, a cycle in the undirected graph $G(\sigma, D)$. This is a contradiction to our assumptions, showing that there is a unique orientation H to satisfy all the (known) in-degree and out-degrees of each vertex. This allows us to reconstruct a unique M from σ, M' and $\sigma(M)'$, given that for all $|D| \leq 2d$, the graph $G(\sigma, D)$ is acyclic.

□

From these two proofs, we can verify a fact stated in the beginning of the paper.

Corollary 3. *A permutation σ on $[n]$ is 1-resilient if and only if consecutive elements in $[n]$ are not consecutive in σ, i.e., $|\sigma(i) - \sigma(i+1)| > 1$ for $1 \leq i < n$.*

Proof. First, we show the forward direction. If σ does map a pair of consecutive elements $(i, i+1)$ in $[n]$ to a pair of consecutive elements $(\sigma(i), \sigma(i+1))$ in σ, we can set $D = \{i, i+1\}$. This yields a 2-cycle (2 edges between the same 2 vertices) in the corresponding $G(\sigma, D)$, so by Theorem 1, σ is not 1-resilient.

In the reverse direction, we assume that σ doesn't map consecutive elements to consecutive elements. Then, for every $D \subset [n]$, where $|D| = 2$, there will at least be two elements in $\{t_1, \ldots, t_k\}$, or at least two elements in $\{b_2, \ldots, b_l\}$, because no two elements i and j are in the same contiguous block in both the first and second line of the two-line representation of σ. So $G(\sigma, D)$, which has only 2 edges and is connected, will not have a cycle. Then, by Theorem 2, σ is 1-resilient. □

2.2 Constructions

In the second half of their paper, Alon et al. also exhibited an algorithm for constructing d-resilient permutations for arbitrarily large d.

Theorem 4. *For any n and d satisfying $n > 3^{2d}$, there is a d-resilient permutation σ of $[n]$. Such a σ can be found by a polynomial time algorithm (in n).*

Call a graph H an (n, d)-double path graph if it has n vertices, its (undirected) edge set is the disjoint union of two distinct Hamiltonian paths (paths that visit each edge exactly once, and potentially including multi-edges), and its girth (the length of the shortest cycle in H) is at least $2d+1$. We can number the vertices of H according to the ordering of the first Hamiltonian path, which corresponds to the top row of the two-line representation of σ. The ordering of the second Hamiltonian path then corresponds to the bottom row of the two-line representation of σ. This forms a correspondence between Hamiltonian double path graphs on n vertices and permutations σ on $[n]$.

For example, for the earlier $\sigma = \binom{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9}{7\ 4\ 1\ 8\ 5\ 2\ 9\ 6\ 3}$ on [9], we can construct the Hamiltonian double path graph as in Figure 8.

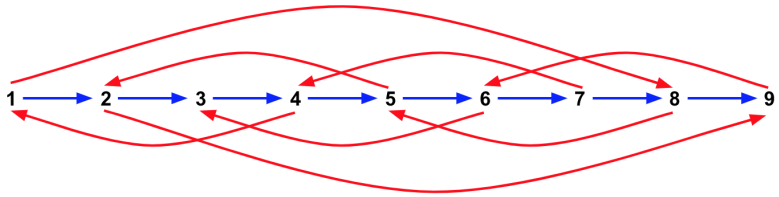

Figure 8: The Hamiltonian double path graph for $\sigma = \binom{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9}{7\ 4\ 1\ 8\ 5\ 2\ 9\ 6\ 3}$ on [9]. Strictly speaking, the Hamiltonian double path graph is not directed, but the arrows are included for clarity. Note that the girth of this graph is 4 (and there are many 4-cycles in this graph, for example, $1-8-9-2-1$).

To prove Theorem 4, we will use the two lemmas below.

Lemma 5. *For any (n,d)-double path graph H, the corresponding permutation is d-resilient.*

Lemma 6. *If $n > 3^{2d}$, then there is an (n,d)-double path graph H, that can be found by a polynomial time algorithm (in n).*

These two lemmas directly imply Theorem 4. Their proofs are below.

Proof of Lemma 5. Given H, we first obtain the corresponding σ. Let the top row of the two-line representation of σ be $[n]$, label the vertices of the Hamiltonian double path graph from 1 through n according to one path, and let the bottom row be an ordering of the Hamiltonian path in H containing all the edges in $E(H) \setminus \bigcup_{1 \le i < n} \{(i, i+1)\}$. (There are two such orderings; one is the reverse ordering of the other, and we can choose either without loss of generality.)

If there is a $D \subset [n]$ so that $G(\sigma, D)$ contains a cycle, then so does the induced subgraph of H' on the vertex set D. This is because every edge in $G(\sigma, D)$ corresponds to a common element between some t_k and b_l. Going through the edges in the shortest cycle of $G(\sigma, D)$, we can write down a sequence S of adjacent vertices (alternating t_k and b_l). Then we can choose pairs of elements in each $s \in S$ such that each adjacent pair shares one common element, and the first and last pair also share a common element. Each pair is an edge in the induced H' on D, and the pairs form a cycle of length $|D|$.

Since H has girth $2d+1$, however, $G(\sigma, D)$ is acyclic for all $|D| \le 2d$. By Theorem 2, σ is d-resilient. □

Proof of Lemma 6. The idea for this proof is to start with an H that is not an (n,d)-double path graph. Let the edges of H be the union of the Hamiltonian path $\{1, 2, ..., n\}$ and another path P. While there is a cycle of length at most $2d$ in H, we will modify P using an iterative method that ensures P stays a Hamiltonian path. The method terminates in a finite number of steps, leaving us with an H that is a (n,d)-double path graph.

While H contains a cycle of length at most $2d$, let C be a shortest cycle in H. Let e be an arbitrary edge of P, which has to exist because the other Hamiltonian path does not have any cycles in

it. Since every vertex has degree at most 4, beginning with any edge e, we have that there are at most $2 \cdot 3$ edges incident to a vertex that is an endpoint of e, at most $2 \cdot 3^2$ edges that are a distance of 2 away from e, and so on. This yields $1 + \sum_{i=1}^{2d-1} 2 \cdot 3^i = 3^{2d} - 2 < n - 1$ edges in H that are within a distance of $2d - 1$ of e (e included).

As a result, P contains an edge e' that is at least a distance $2d$ away from e. We now remove edges e and e' from H, and then add edges to keep P a Hamiltonian path. For instance, letting $e = (\sigma(i), \sigma(i+1))$ and $e' = (\sigma(j), \sigma(j+1))$ where $i < j$, we could add the edges $(\sigma(i), \sigma(j))$ and the edges $(\sigma(i+1), \sigma(j+1))$ so the modified P is still Hamiltonian. Then C is eliminated.

We consider new cycles that were created by the process. There are two cases. If a new cycle contains one of the new edges, then its length is at least $2d + 1$ because the two vertices in a new edge are at least a distance $2d$ apart when the edge between them is excluded. Secondly, consider the case where a new cycle contains both of the new edges. In the old H, we know that the shortest cycle containing $\sigma(i)$ and $\sigma(i+1)$ has length at least $|C|$, and the shortest cycle containing $\sigma(j)$ and $\sigma(j+1)$ also has length at least $|C|$. These two cycles cannot share a common edge, because e and e' are at least a distance $2d$ apart. As a result, in the new H, any new cycle containing the new edges, and therefore the vertices $\sigma(i)$, $\sigma(i+1)$,$\sigma(j)$, and $\sigma(j+1)$, has length at least twice of $|C|$. Thus, the length of the shortest cycle increases after finitely many steps, where each step consists of the above modification to P.

Since the number of cycles of length t in a graph of maximum degree 4 and n vertices is smaller than $n \cdot 3^t$ (choose one of the n vertices to start, and then you can choose from at most 3 edges $t - 1$ times), the process terminates after at most $O(n(3^2 + ... + 3^{2d})) = O(n^2)$ steps.

We lastly check that this is polynomial time in total because finding the shortest cycle in a graph can be done in polynomial time with respect to the number of vertices and edges in the graph. For example, one such algorithm (which may not be optimal) would be to run breadth-first-search sequentially, starting at each vertex of the graph. A breadth-first-search starting at vertex v would be able to detect the shortest cycle that contains v (by terminating the first time it detects a path from v to a neighbor of v). A breadth-first-search runs in time $O(|V| + |E|)$, so finding the shortest cycle in the entire graph would take $O(|V| \cdot (|V| + |E|))$ without optimization. In this case, $|V| = n$ and $|E| = 2(n-1)$, so finding the shortest cycle takes $O(n^2)$, and the algorithm to find a (n, d)-double path graph H takes $O(n^4)$ overall. □

For any n and d satisfying $n > 3^{2d}$, we know know that there is a d-resilient permutation σ of $[n]$. To see that this is the best possible (asymptotically), we have the theorem below.

Theorem 7. *If there is a permutation σ of $[n]$ that is d-resilient, then $d \leq O(\log n)$.*

Proof. While the proof relies on previous work in [7] based on the average degree of a graph, we will use a similar method based on the minimum degree of a graph [8]. Given the minimum degree of a graph and its girth, we can find a lower bound on the number of vertices in the graph. As a result, given a graph on n vertices with a minimum degree, we can upper-bound its girth, and thus obtain an upper bound on d.

Given the minimum degree of a graph Δ and girth $2d+1$, we can use the Moore graph construction to get a lower bound on the number of vertices in the graph. (The construction is similar to a technique we did in class on extremal graph theory for the Turán problem.) Starting with any vertex in the graph, we construct a tree: There are at least Δ vertices adjacent to the starting vertex, and at least $\Delta - 1$ distinct vertices adjacent to each of those vertices so as to not form a cycle

of length less than $2d+1$, and so on. This method yields a lower bound, $1+\Delta\sum_{i=0}^{d-1}(\Delta-1)^i < n$. (Similarly, for even girth $2d$, we can construct a tree by starting at an edge. Then each of the two vertices have at least $\Delta-1$ distinct vertices adjacent to it, and we can keep adding layer-like sets of $\Delta-1$ distinct vertices $d-1$ times. This yields $2\sum_{i=0}^{d-1}(\Delta-1)^i < n$.)

From a d-resilient σ, we form the corresponding (n,d)-double path graph, which has minimum degree $\Delta = 3$. Then we have that $1+3(2^d) < n$, so $d \leq O(\log n)$.

□

3 Extensions

Having summarized Alon et al.'s article, we now look into some possible extensions of their work, in various ways.

3.1 Algorithm to Reconstruct M from M' and $\sigma(M)'$

Suppose that we have a d-resilient permutation σ on $[n]$, for which we receive M' and $\sigma(M)'$ for some M that our friend has sent us. In this paper, Alon et al. has demonstrated that there is a unique M consistent with both M' and $\sigma(M)'$. However, the proof only shows uniqueness. A natural follow-up question would be whether an efficient algorithm exists to actually reconstruct M from M' and $\sigma(M)'$, and how fast this algorithm would run. Here, we show that $O(nd+d^3)$ algorithm does exist, using a network flow method.

Following the proof of Theorem 2, we first determine the location of all doubly-occurring symbols (that are present in both M' and $\sigma(M)'$). Since there are n symbols in M and n symbols in $\sigma(M)$, and d distinct symbols are deleted from each, there are $n-2d$ doubly-occurring symbols.

Recall that for each doubly-occurring symbol x, we generate d possible indices of x in $\sigma(M)$, given by $\{a_0, a_1, \ldots, a_d\}$, with $a_i := \sigma^{-1}(k+i)$ where k is the index of x in M'. For each a_i, we check whether the actual index of x in $\sigma(M)'$ is within the set $S_i := \{a_i, a_i-1, \ldots, a_i-d\}$, which takes $O(1)$ time. Hence, the position of x in M can be determined in $O(d)$ time overall. With $n-2d$ doubly-occurring symbols, we can find the locations of all doubly-occurring symbols in $O(d(n-2d)) = O(nd)$ time.

Now, given the locations of all doubly-occurring symbols, we can set D to be the (complement) set of singly-occurring symbols, and generate $G = G(\sigma, D)$ in $O(n)$ time, since G has at most $2n$ vertices and n edges.

As argued in Theorem 2, since we receive M' and $\sigma(M)'$, we know the exact supposed in-degree and out-degree of each vertex in G. In addition, by the condition of d-resilience, we know that G is acyclic. It suffices to devise an algorithm to assign an orientation H to all the edges of G such that all the in-degree and out-degrees are satisfied (we have shown that such a H is unique in Theorem 2, and such a H exists because this graph is generated from a genuine M that your friend has sent us).

To build intuition, an example of how the graph looks right now is in Figure 9:

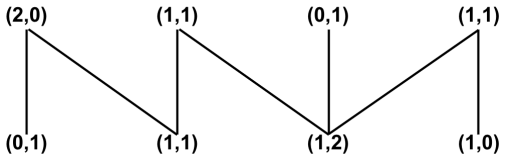

Figure 9: A bipartite acyclic graph G where we know the supposed in-degrees and out-degrees of each vertex. This is denoted (in, out) for each vertex. We want to devise an algorithm to assign an orientation H to this graph to satisfy all these (in, out) pairs.

We claim that a simple network flow algorithm can do this assigning of orientation H to G. We create the network flow via the following steps:

- For each undirected edge in G, convert that edge into two edges pointing in opposite directions, each with weight 1 (the black edges in Figure 10).
- Create a source vertex s and a sink vertex t. For each vertex v, compute the quantity $v_{out} - v_{in}$. If this quantity is positive, connect an edge from s to v with weight $v_{out} - v_{in}$. If this quantity is negative, connect an edge from v to t with weight $v_{in} - v_{out}$. If this quantity is 0, do not add any edges.

The directed network flow graph now looks like Figure 10 (where each black edge has weight 1, omitted for clarity).

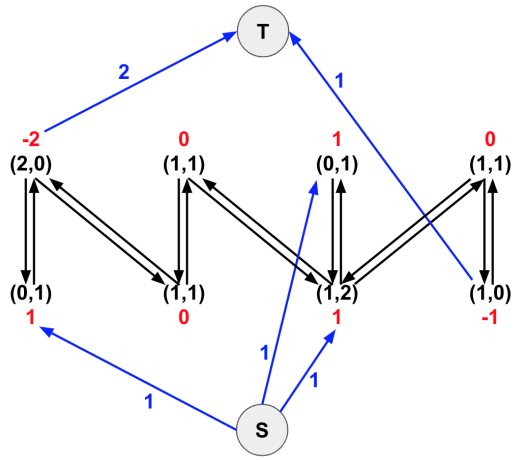

Figure 10: The network flow graph generated from the graph G above.

We know that there are efficient network flow algorithms to compute the maximum flow from s to t in this graph. For example, using the Edmonds-Karp optimization of the Ford-Fulkerson algorithm, we can ensure find the maximum flow in $O(|V||E|^2)$ [9]. Based on the construction of G, $|V| \leq 2d$ and $|E| = d$. The maximum flow can be thus found in $O(d^3)$.

Before we prove that the maximum flow from s to t suffices to solve for an orientation H, we calculate the total runtime of this reconstruction algorithm. The locations of all doubly-occurring symbols can be determined in $O(nd)$, and the orientation of $G(\sigma, D)$ can be determined in $O(d^3)$, which allows us to determine the locations of all singly-occurring symbols and recover M. The total run time of the reconstruction algorithm is thus only $O(nd + d^3)$, which is pretty efficient!

Now, it suffices to show that the maximum flow indeed corresponds to an orientation H on the graph. First, because we know that a valid orientation H exists because G is generated from a genuine M, the maximum flow found will always be the sum of the out-weights of s, and the sum of the sum of the in-weights of t, as in Figure 11.

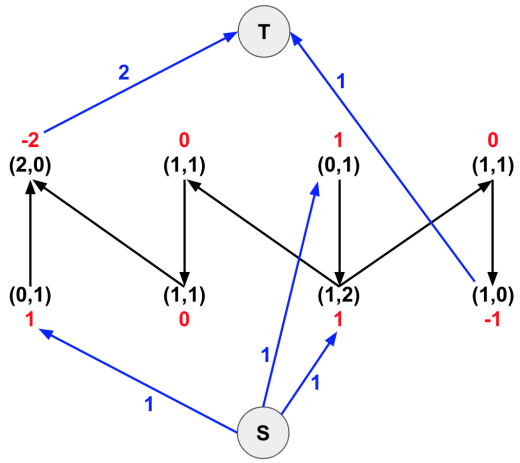

Figure 11: The network flow graph with maximum flow.

Now, the only possible complication for such a maximum flow to not correspond to an orientation H of the edges, is if at least one black edge is not used in the maximum flow (or both the edge and its back-edge are used, which is equivalent). By way of contradiction, suppose we have a maximum flow that does this. This is depicted by the gray edge in 12.

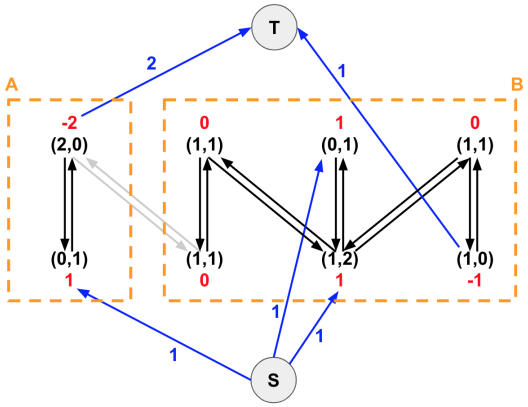

Figure 12: The only possible case where an edge is not used in a maximum flow.

However, because G is acyclic, this would split the graph G up into two connected components, labelled A and B in Figure 12. If we just consider the flows into and out of the connected components A and B, as in Figure 13, we realize a problem.

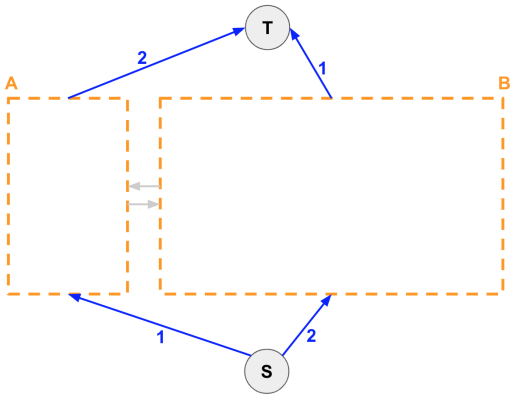

Figure 13: Abstracting the connected components A and B for the flow.

Obviously, because a valid orientation H exists which utilizes exactly one of the gray edges, the flow into (from s) and out of (to t) the connected component A will not be equal. This means that any flow that does not utilize both gray edges, will not be using the maximum capacity out of s or into t. Such a flow cannot be a maximum flow, since we know the maximum flow to have the value of the sum of the out-weights of s, and the value of the sum of the in-weights of t. This is a contradiction and confirms that any maximum flow will utilize exactly one of each black edge, therefore corresponding to an orientation H and concluding our proof that the reconstruction algorithm is only $O(nd + d^3)$.

3.2 What if the Same Symbol is Deleted from M and $\sigma(M)$?

Recall that Alon et al. assumed that the $\leq d$ symbols deleted from M to produce M' and the $\leq d$ symbols deleted from $\sigma(M)$ to produce $\sigma(M)'$ are disjoint for their further analysis. We denote these two sets of symbols as $del(M)$ and $del(\sigma(M))$ for brevity. Of course, their assumption makes sense, because when they are not disjoint, i.e. when $|del(M) \cap del(\sigma(M))| > 0$, we have no way of knowing the values of symbols in $del(M) \cap del(\sigma(M))$, which don't show up in either M' or $\sigma(M)'$.

However, even if the same symbol(s) is deleted from both, it may be the case that we can still find the position of the deleted symbol(s) in M from M' and $\sigma(M)'$. Equivalently, this means that, in the specific case where we are sending a message of n distinct symbols from an alphabet with only n distinct symbols (i.e. the message is effectively a permutation of $[n]$), we can reconstruct M from M' and $\sigma(M)'$, even if $|del(M) \cap del(\sigma(M))| > 0$.

We will term a permutation σ on $[n]$ as *d-(k-overlap)-resilient* iff M can be reconstructed from M' and $\sigma(M)'$, given that M is a permutation of $[n]$, $|del(M)|, |del(\sigma(M))| \leq d$, and $|del(M) \cap del(\sigma(M))| \leq k$.

This generalizes d-resilience, which is equivalent to d-(0-overlap)-resilient, by this definition. Note also that $d \geq k$ for this definition to make sense. We will proceed to prove the following Lemmas.

Lemma 8. *Any 1-resilient permutation σ on $[n]$ is also 1-(1-overlap)-resilient.*

Lemma 9. *For $d \geq k \geq 2$, no σ on $[n]$ is d-(k-overlap)-resilient.*

Lemma 10. *For $d > 1$, in general, σ being d-resilient does not imply σ is d-(1-overlap)-resilient.*

Proof of Lemma 8. This is fairly straightforward. We know that $|del(M)|, |del(\sigma(M))| \leq 1$, and that $|del(M) \cap del(\sigma(M))| \leq 1$, so the only deviation from a standard 1-resilience proof is when $|del(M) \cap del(\sigma(M))| = |del(M)| = |del(\sigma(M))| = 1$, i.e. the same symbol is deleted from both M and $\sigma(M)$. In this case, the positions of all $n-1$ doubly occurring symbols can still be determined by the method in Theorem 2, so we can uniquely determine the position of the missing symbol in M. Given that M is a permutation of $[n]$, we can reconstruct M. □

Proof of Lemma 9. This is again fairly straightforward. Given that $|del(M) \cap del(\sigma(M))| \leq 2$, we can construct the worst case scenario where $|del(M) \cap del(\sigma(M))| = 2$. Suppose without loss of generality that the symbols '1' and '2' are deleted from the message M (which is a permutation of $[n]$), and are also deleted from the message $\sigma(M)$, and we receive M' and $\sigma(M)'$.

We can easily construct an alternative $\hat{M}$ which is identical to M except that the positions of '1' and '2' are swapped. By deleting both '1' and '2' from $\hat{M}$ and $\sigma(\hat{M})$ (among other symbols, perhaps), we will be unable to distinguish M from $\hat{M}$ when receiving M' and $\sigma(M)'$.

This suffices to show that no σ on $[n]$ is d-(k-overlap)-resilient when $d \geq k \geq 2$. □

Proof of Lemma 10. Observe that the first half of the algorithm to reconstruct M given M' and $\sigma(M)'$ is still valid even if $|del(M) \cap del(\sigma(M))| > 0$. That is, since σ is d-resilient, $G(\sigma, D)$ is still acyclic for all $|D| \leq 2d$, so we are still able to determine the location of all doubly-occurring symbols that are present in M' and $\sigma(M)'$.

This means that we are still able to reconstruct a figure that looks like Figure 7. However, the complication that $|del(M) \cap del(\sigma(M))| > 0$ poses us is that a valid deletion of $\leq d$ symbols from M and $\leq d$ symbols from $\sigma(M)$ no longer corresponds to a simple orientation of the edges in Figure 7. In particular, if the same symbol '4' is deleted from both M and $\sigma(M)$, then the following generalized-orientation (gen-orientation) in Figure 14 is possible.

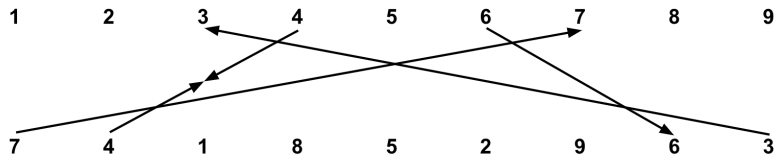

Figure 14: A gen-orientation of $G(\sigma, D)$ that corresponds to a deletion of the same symbol '4' from both M and $\sigma(M)$. Note that the edge between the two 4's are out-edges on both ends, denoting precisely that deletion. We will term such edges *squish edges*.

Based on this formulation, it is clear that M can be reconstructed from M' and $\sigma(M)'$ iff there is only one gen-orientation that can be assigned to edges in $G(\sigma, D)$ that satisfies all the (known) in-degree and out-degrees of each vertex. Specifically, a gen-orientation means assigning edges to be in either direction, or to be a squish edge.

Figure 15 is an example of a acyclic $G(\sigma, D)$ that permits multiple gen-orientations with only 1 squish edge each, proving our general claim that d-resilience does not imply d-(1-overlap)-resilience. Note that $G(\sigma, D)$ being acyclic already implies that σ is d-resilient.

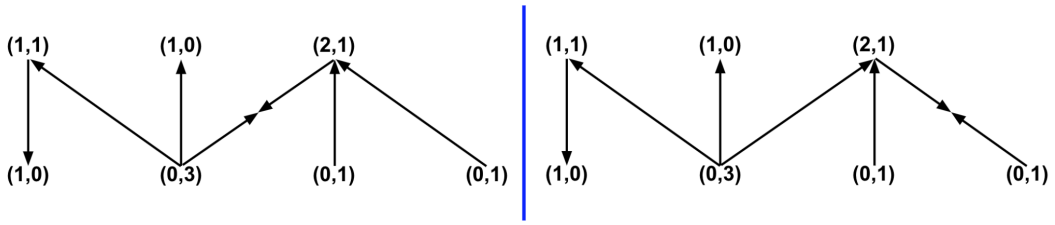

Figure 15: An example of an acyclic $G(\sigma, D)$, together with specified in-degree and out-degrees of each vertex, that permits multiple gen-orientations with only 1 squish edge each. It is worth noting that this statement is not true if we allow only orientations instead of gen-orientations (as proved earlier).

□

We have thus shown that no σ on $[n]$ is d-(k-overlap)-resilient when $d \geq k \geq 2$. When $k = 1$ and $d = 1$, 1-resilience implies 1-(1-overlap)-resilience. But, when $k = 1$ and $d > 1$, we showed that d-(1-overlap)-resilience is a stronger condition than d-resilience.

It is left as an open question whether there are d-(1-overlap)-resilient permutations for $d > 1$, and if so, how they can be characterized.

3.3 Repeated Symbols in M

In an arbitrary message, it is reasonable to imagine there are repeated symbols; for example, if we wish to send the word "combinatorics", there are several repeated letters. It is natural to wonder whether the permutation methodology described here extends to this slightly more practical setting. Unfortunately, they do not.

Lemma 11. *If we consider messages M of length n with possibly repeated symbols, then there are no permutations σ on $[n]$ that are even 1-resilient.*

Proof of Lemma 11. We will construct two messages M_1 and M_2 such that, for any permutation σ, we can delete one symbol from each of M_1, $\sigma(M_1)$, M_2, and $\sigma(M_2)$ to yield $M_1' = M_2'$ and $\sigma(M_1)' = \sigma(M_2)'$.

Let $M_1 = AAA\cdots AAB$ and $M_2 = AAA\cdots ABA$. In words, M_1 has B in the n^{th} position and A in all others; M_2 has B in the $(n-1)^{th}$ position and A in all others. Let σ be an *arbitrary* permutation; then $\sigma(n) \neq \sigma(n-1)$. In M_1 delete the first A, and in M_2 the last A; in $\sigma(M_1)$ and $\sigma(M_2)$ delete the B. This is shown below for a particular permutation σ (but the same idea holds for an arbitrary permutation).

$$\begin{aligned} M_1 &= AAA\cdots AAB & M_2 &= AAA\cdots ABA \\ M_1' &= \cancel{A}AA\cdots AAB & M_2' &= AAA\cdots AB\cancel{A} \\ \sigma(M_1) &= AAB\cdots AAA & \sigma(M_2) &= AAA\cdots BAA \\ \sigma(M_1)' &= AA\cancel{B}\cdots AAA & \sigma(M_2)' &= AAA\cdots \cancel{B}AA \end{aligned}$$

Then $M_1' = M_2' = AAA \cdots AAB$ and $\sigma(M_1)' = \sigma(M_2)' = AAA \cdots AAA$, so the two messages are indistinguishable. Given M_1' and $\sigma(M_1)'$ it is not possible to reconstruct any message.

□

Of course, allowing for an arbitrary number of repeated characters is a fairly strong assumption. If we make the much weaker assumption that there's a single repeated character in the message, which occurs only twice, can we have 1-resilient permutations? Here, a completely general counterexample is less obvious, so we begin with $n = 4$. Observe that, by Corollary 3, a permutation is 1-resilient iff it separates adjacent symbols (i.e. if A and B are adjacent in M, they must not be adjacent in $\sigma(M)$). Observe that in $[1, 2, 3, 4]$, 2 is adjacent to two other elements, so it must be mapped to the first or last positions by σ, otherwise, it would be adjacent to two symbols in σ as well, and there must be some overlap between the symbols it's adjacent to originally (1 and 3) and in σ. By the same reasoning, 3 must occur in the first or last position. With this in mind, the only 1-resilient permutations for $n = 4$ are $\sigma_1 = \binom{1\ 2\ 3\ 4}{2\ 4\ 1\ 3}$ and $\sigma_2 = \binom{1\ 2\ 3\ 4}{3\ 1\ 4\ 2}$. But we construct the following counterexample for σ_1:

$$\begin{aligned} M_1 &= ABAC & M_2 &= ABCA \\ M_1' &= AB\not{A}C & M_2' &= ABC\not{A} \\ \sigma_1(M_1) &= BCAA & \sigma_1(M_2) &= BAAC \\ \sigma_1(M_1)' &= B\not{C}AA & \sigma_1(M_2)' &= BAA\not{C} \end{aligned}$$

And, since σ_2 simply reverses σ_1, the reverse of the same counterexample also shows that σ_2 is not 1-resilient for even one repeated symbol.

Using a Matlab program, we computed all 1-resilient permutations and found a counterexample (distinct M_1 and M_2 for which $M_1' = M_2'$ and $\sigma(M_1) = \sigma(M_2)$). This was done for the 14 1-resilient permutations for $n = 5$ (examples of which are in the Appendix, Section 4), 90 1-resilient permutations for $n = 6$, and 646 1-resilient permutations for $n = 7$. ($n = 8$ became computationally infeasible.) As such, we conjecture that there are no 1-resilient permutations with even 1 repeated symbol, but leave as an open question explicitly constructing these counterexamples for arbitrary permutations.

3.4 Deleting Different Numbers of Elements from M and $\sigma(M)$

Given a d-resilient permutation, what if we delete a total of $2d$ distinct elements, lifting the restriction that d must be deleted from each of M and $\sigma(M)$? For example, if we deleted one element from M and $2d - 1$ from $\sigma(M)$, M is definitely recoverable.

We conjecture that the deletion of d elements from both is a "worst case", analogous to a convexity argument. This is left as an open question as well.

3.5 Runtime of Checking d-Resilience

Alon et al. have provided a necessary and sufficient condition for σ to be d-resilient, namely, σ is d-resilient iff for all $D \subset [n]$ such that $|D| \leq 2d$, $G(\sigma, D)$ is acyclic. Here, we investigate whether

this is practically meaningful. In particular, is there is a difference in algorithmic runtime between directly checking the d-resilience of σ, and checking whether $G(\sigma, D)$ is acyclic for all $|D| \leq 2d$?

First, we discuss how to check d-resilience of σ using LCS methods. If σ is d-resilient, then there exists no M_1 and M_2 where some $l_1 = LCS(M_1, M_2)$ and $l_2 = LCS(\sigma(M_1), \sigma(M_2))$ such that $|l_1| = |l_2| = n - d$ and $l_1^c \cap l_2^c = \emptyset$ ($l_1^c = [n] \setminus l_1$, if we treat l_1 as a set). Otherwise, we would be able to delete l_1^c from each of M_1 and M_2 to get $M_1' = M_2' = l_1$, and l_2^c from each of $\sigma(M_1)$ and $\sigma(M_2)$ to get $\sigma(M_1)' = \sigma(M_2)' = l_2$.

This allows us to develop a straightforward method of checking whether a σ is d-resilient. We first fix $M_1 = [n]$, and then we find all the possible M_2 such that $LCS(M_1, M_2) = d$. We can estimate the number of such M_2, $\#M_2 \leq \binom{n}{d}\binom{n}{d}d! \approx n^{2d}d^d$. This can be derived by choosing a particular subsequence in M_1, then choosing their positions in M_2, and then arranging the remaining d elements. The inequality results from some permutations M_2 being double-counted. Checking that a given M_2 satisfies $LCS(\sigma(M_1), \sigma(M_2)) = d$ takes $O(n^2)$ time via dynamic programming, so the runtime to check whether a σ is d-resilient is approximately $O(n^{2d+2}d^d)$.

We can compare this to the runtime of checking whether σ is d-resilient by constructing all possible $G(\sigma, D)$. There are $\binom{n}{2d} \approx n^{2d}$ ways of choosing D, and checking whether $G(\sigma, D)$ has a cycle for each D takes $O(d)$, since a simple breadth-first search would suffice, giving a total runtime of $O(n^{2d}d)$.

This is on a similar order to the LCS estimate (the d^d term can probably be shaved off with a tighter upper bound on $\#M_2$), which indicates that this necessary and sufficient condition (while mathematically insightful) is not really practically meaningful.

3.6 Finding σ for Every n

The construction gives us a σ for every d. But we've also seen examples of small σ that are d-resilient for some d where $n \leq 3^{2d}$, and these cannot be constructed using the given method. For example, we have a 1-resilient σ (beginning of paper) in which $n = 4$, but the construction gives us 1-resilient σ when n is at least 10. Similarly, the paper says that the smallest 2-resilient σ is for $n = 18$, but the construction only works once n is at least 82. So another natural question is, how could we construct σ for a given n, for as many d as possible?

This is left as an open question.

3.7 Number of σ as a Function of n and d

We can also ask how many distinct d-resilient σ can be constructed for each n. Let us denote this number as $f_d(n)$.

For example, for $d = 1$, it is a well-researched problem to find the number of distinct 1-resilient σ's for each n, or equivalently, to find the number of permutations on $[n]$ that do not have any rising or falling contiguous subsequences of length 2.

For instance, this is investigated in [10]. The result shows that we know the following recurrence:

$$f_d(0) = f_d(1) = 1, \quad f_d(2) = f_d(3) = 0$$

$$f_d(n) = (n+1)f_d(n-1) - (n-2)f_d(n-2) - (n-5)f_d(n-3) + (n-3)f_d(n-4)$$

This is also found as a sequence on OEIS. It is left as an open question whether there is a expression for this sequence for larger d.

3.8 Practical Comparison Between Error-Correcting Codes

We now put two of the error-correcting codes in conversation with each other from a communication perspective, beginning with the main focus of the work, deletion-resilient permutations [1]. In this scheme, for communication we only really need one deletion-resilient permutation. Once we have that permutation, we can apply it to every n-length message. We require sending $2n$ characters per n-length message, but there is no restriction on which messages you can send - M as any permutation on $[n]$ is allowed. (An interesting concern is making sure M and $\sigma(M)$ are distinguishable, but here we assume they can be sent separately without issue and simply add the lengths.) The construction in of Alon et al. demonstrated that for $n > 3^{2d}$, there is a d-resilient permutation of $[n]$; asymptotically, $d < O(\log(n))$. So it would seem that for fixed d and an $n > 3^{2d}$, we can send $2n$ characters (in two segments) across a d-deletion channel for any of the $n! = (3^{2d})!$ messages, and reconstruct the original message. Via Section 3.1, the decoding algorithm takes time $O(nd+d^3) = O(d3^{2d})$. However, there is a caveat - the coding scheme only works when the deletions are disjoint between M and $\sigma(M)$. To quantify this, let us assume that a set of d deletions are chosen uniformly from $\binom{n}{[d]}$. Then for any set of d deletions in the transmission of M, there is only a $\sim \left(\frac{d-1}{d}\right)^d$ chance that a disjoint set of symbols will be deleted from the second message (where we use the approximation that each symbol is deleted independently with probability $\frac{1}{d}$).

The longest common subsequence formulation takes a slightly different approach. Here, we consider a restricted set of n-length codewords, i.e. message family of size $< n!$. We consider a family of permutations of longest (pairwise) common subsequence of length s. Then, only one of these codewords is sent; the recipient computes the longest common subsequence between the received word (with deletions), and the predetermined family of codewords. So long as $d < n-s$ codewords were deleted, selecting the message with the longest common subsequence with the received word is guaranteed to yield the correct original message. How large a family of codewords can we hope to find? Beame, Blais, and Huynh-Ngoc showed that for $k \geq 3$ and $n \geq k^2$ we can find a family of k permutations of $[n]$ such that the longest common subsequence is at most $32\sqrt[3]{kn}$ [6]. Previously, the known lower bound was that any family of ≥ 3 permutations of $[n]$ had a longest common subsequence of length at least $n^{1/3}$. With this in mind, we know there exists a family of $k = \sqrt{n}$ permutations of $[n]$ with LCS $\leq 32\sqrt[3]{kn} = 32\sqrt[3]{kn} = 32\sqrt{n}$. Decoding requires making $\binom{k}{2} = \binom{\sqrt{n}}{2}$ LCS comparisons, each of which has complexity $O(n^2)$. Overall, the decoding complexity is $O(n^3)$. However, the probability of success is 1: decoding is guaranteed to be successful regardless of which characters were deleted, so long as $d < n - s = n - 32\sqrt{n}$.

Between these two schemes, one is not clearly better or worse. Deletion-resilient permutations allow for a far larger set of messages to be sent, $n! = (3^{2d})!$ as opposed to $\sqrt{n} \sim \sqrt{d}$ for LCS (using the very rough approximation between d and n for LCS). However, LCS sends much shorter messages, of length $n \sim d$ as opposed to $2n = 2 \cdot 3^{2d}$, and succeeds with probability 1 instead of $\left(\frac{d-1}{d}\right)^d$. Finally, decoding is faster for LCS than deletion-resilient permutations, at least as a function of d; as a function of n, both are polynomial in n.

Method	n and d Relation	Input Family	Encoding Size	$\mathbb{P}$(success)	Runtime	Source
Permutations	$n > 3^{2d}$	$n!$	$2n$	$\left(\frac{d-1}{d}\right)^d$	$O(d3^{2d})$	[1]
LCS	$d < n - 32\sqrt{n}$	$\sqrt{n}$	n	1	$O(n^3)$	[6]

Figure 16: A quantitative comparison between various encoding methods as practical error-correcting schemes.

4 Appendix

Here we include counter-examples generated for a few 1-resilient permutations on $n = 5$.

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 3 & 1 & 4 & 2 \end{pmatrix}$$

$$\begin{aligned} M_1 &= DCBAA & M_2 &= DCAAB \\ M_1' &= DCB\not{B}AA & M_2' &= DCAA\not{B} \\ \sigma_1(M_1) &= ABDAC & \sigma_1(M_2) &= BADAC \\ \sigma_1(M_1)' &= \not{A}BDAC & \sigma_1(M_2)' &= B\not{A}DAC \end{aligned}$$

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 1 & 5 & 3 \end{pmatrix}$$

$$\begin{aligned} M_1 &= DCBAA & M_2 &= DCAAB \\ M_1' &= DCB\not{B}AA & M_2' &= DCAA\not{B} \\ \sigma_1(M_1) &= CADAB & \sigma_1(M_2) &= CADBA \\ \sigma_1(M_1)' &= CAD\not{A}B & \sigma_1(M_2)' &= CADB\not{A} \end{aligned}$$

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 1 & 4 & 2 \end{pmatrix}$$

$$\begin{aligned} M_1 &= DCAAB & M_2 &= DAACB \\ M_1' &= D\not{C}AAB & M_2' &= DAA\not{C}B \\ \sigma_1(M_1) &= ABDAC & \sigma_1(M_2) &= ABDCA \\ \sigma_1(M_1)' &= ABD\not{A}C & \sigma_1(M_2)' &= ABDC\not{A} \end{aligned}$$

References

[1] Noga Alon, Steve Butler, Ron Graham, and Utkrisht C Rajkumar. Permutations resilient to deletions. *Annals of Combinatorics*, 22(4):673–680, 2018.

[2] Error correcting codes. `https://plus.maths.org/content/error-correcting-codes`, Oct 2018.

[3] Harry Fairhead. How error correcting codes work. `https://www.i-programmer.info/babbages-bag/214-error-correcting-codes.html`, Mar 2018.

[4] Error detection and correction. https://en.wikipedia.org/wiki/Error_detection_and_correction#Error-correcting_code, Apr 2019.

[5] Neil JA Sloane. On single-deletion-correcting codes. *Codes and designs*, 10:273–291, 2000.

[6] Paul Beame, Eric Blais, and Dang-Trinh Huynh-Ngoc. Longest common subsequences in sets of permutations. *arXiv preprint arXiv:0904.1615*, 2009.

[7] Noga Alon, Shlomo Hoory, and Nathan Linial. The moore bound for irregular graphs. *Graphs and Combinatorics*, 18(1):53–57, 2002.

[8] Mirka Miller and Jozef Sirán. Moore graphs and beyond: A survey of the degree/diameter problem. *The Electronic Journal of Combinatorics*, 1000:DS14–Dec, 2005.

[9] Jack Edmonds and Richard M Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.

[10] Morton Abramson and WOJ Moser. Permutations without rising or falling ω-sequences. *The Annals of Mathematical Statistics*, 38(4):1245–1254, 1967.