JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 0, Number 0, 2017

Pp. 1–16
DOI: 10.1089/cmb.2017.0060

Research Article

A Linear-Time Algorithm for the Copy Number Transformation Problem

RON ZEIRA,[1,*] MEIRAV ZEHAVI,[2,*] and RON SHAMIR[1]

ABSTRACT

Problems of genome rearrangement are central in both evolution and cancer. Most evolutionary scenarios have been studied under the assumption that the genome contains a single copy of each gene. In contrast, tumor genomes undergo deletions and duplications, and thus, the number of copies of genes varies. The number of copies of each segment along a chromosome is called its copy number profile (CNP). Understanding CNP changes can assist in predicting disease progression and treatment. To date, questions related to distances between CNPs gained little scientific attention. Here we focus on the following fundamental problem, introduced by Schwarz et al.: given two CNPs, *u* and *v*, compute the minimum number of operations transforming *u* into *v*, where the edit operations are segmental deletions and amplifications. We establish the computational complexity of this problem, showing that it is solvable in linear time and constant space.

Keywords: copy number, edit distance, genome rearrangement.

1. INTRODUCTION

The genome of a species evolves by undergoing small and large mutations over generations. Large mutations modify genome organization by rearrangement of genomic segments. Computational analysis of the process of *genome rearrangement* has been the subject of extensive research over the last two decades (Fertin et al., 2009). The majority of these studies to date were restricted to a single copy of each gene and were concerned with the reordering of segments. Extant models that do not make this assumption often result in NP-hard problems (Tannier et al., 2009; Savard et al., 2011; Shao and Lin, 2012).

While most work on genome rearrangements to date was done in the context of species evolution, there is today great opportunity in analysis of cancer genome evolution. Cancer is a dynamic process characterized by the rapid accumulation of somatic mutations, which produce complex tumor genomes. Species evolution happens over eons and changes are carried over from one generation to the next. In contrast, cancer evolution happens within a single individual over a few decades. In many tumor genomes, a lot of the changes are segmental deletions and amplifications (The Cancer Genome Atlas Research Network, 2011). As a result, the number of copies of each segment along a chromosome, known as its copy number profile (CNP), changes during cancer development, compared to the normal genome that has two copies

[1]Blavatnik School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel.
[2]Department of Informatics, University of Bergen, Bergen, Norway.
*These authors contributed equally to this work.

(or *alleles*) for each segment. Understanding these changes can assist in predicting disease progression and the outcome of medical interventions. However, computational questions related to distances between CNPs received little scientific attention to date. Such questions are the topic of this article.

Over the years, a variety of methods were used to determine the CNP of a cancer genome, at different resolutions. G-banding allows viewing the chromosome bands (Pinkel et al., 1986). Fluoroscent In Situ Hybridization (FISH) measures the copy numbers of tens to hundreds of targeted genes (Chowdhury et al., 2014). Array comparative genomic hybridization gives a higher resolution of CN estimation for a cell population (Urban et al., 2006). Most recently, deep sequencing techniques yield CNPs by using read depth data (Oesper et al., 2012). While it would have been preferable to analyze the genome (karyotype) itself and not its CNP, detection of structural variations from sequencing data is still problematic (McPherson et al., 2012; Abo et al., 2014). Today it is a routine procedure to obtain detailed CNPs of cancer genomes, but utilizing them to understand cancer evolution is still an open problem.

Given two CNPs, the healthy tissue's and the tumor's, evaluating the distance between them can help in understanding cancer progression. A naive measure of distance is the Euclidean distance between the two profiles (Schwarz et al., 2014). Chowdhury et al. defined edit distance between CNPs obtained from FISH, where the edit operations are amplification or deletion of single genes, single chromosomes, or the whole genome (Chowdhury et al., 2013, 2014, 2015). However, calculating these distances requires exponential time in the number of genes and therefore is limited to low-resolution FISH data. The *TuMult* algorithm uses the number of breakpoints (loci where the CNs change) between two profiles as a simple distance measure (Letouzé et al., 2010).

Schwartz et al. introduced a model that admits amplification and deletion of contiguous segments (Schwarz et al., 2014). The edit distance between two CNPs was defined as the minimum number of segmental deletions and duplications over all separations of the profiles into two alleles (a procedure known as *phasing*). Their algorithm *MEDICC* for computing the edit distance uses finite-state transducers (FSTs) (Mohri, 2003) to model the profiles and efficiently compute the distance. However, the complexity of this method was not analyzed. Even without the phasing computation, the method needs to compose a three-state transducer with itself B times, resulting in a transducer with 3^B states (Mohri, 2004; Schwarz et al., 2014). Here, B is the maximum CN in the input. The running time of FST procedures relies on the number of states and transitions, and in some cases may be exponential (Mohri, 2003, 2004).

1.1. Copy number transformation

We investigate the following problem, which underlies the model of Schwarz et al. (2014): Given two CNPs, u and v, compute the minimum number of segmental duplications and deletions needed to transform u into v. We call this problem the Copy Number Transformation Problem (CNTP). A CNP is represented by a vector of non-negative integers (the number of copies of each segment). A segmental deletion (amplification) decreases (resp. increases) by 1 the values of a contiguous interval of the vector, where zero values are not affected. Formal definitions are given in Section 2.

1.2. Our contribution

We show that the CNTP is solvable in linear time and constant space. The algorithm relies on several properties of the problem that we establish in Section 3.1, which may also be relevant to the analysis of other problems involving CNPs. By exploiting these properties, we obtain a pseudopolynomial dynamic programming algorithm for CNTP, presented in Section 3.2. In Section 3.3, by establishing that a certain function in the dynamic programming recursion is piecewise linear, we improve its performance and obtain our main result, namely, a linear-time algorithm for CNTP.

Preliminary version of this article appeared in the proceedings of CPM 2016 (Shamir et al., 2016).

2. PRELIMINARIES

In this section, we give definitions and notations that are used throughout the article. Let $n \in \mathbb{N}$. A CNP is a vector $V = (v_1, v_2, \ldots, v_n)$, where $v_i \in \mathbb{N} \cup \{0\}$. Each position in V corresponds to a segment in the normal genome, where the segments are ordered as in the normal genome. For simplicity we call a position a *gene*. A CN operation (CNO) is a triple $c = (\ell, h, w)$, where $1 \leq \ell \leq h \leq n$ and $w \in \{-1, 1\}$. We say that

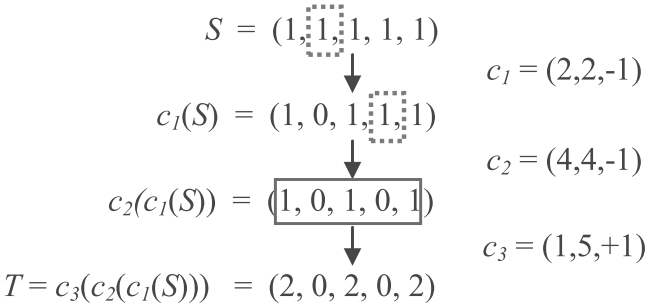


FIG. 1. The CNT $C=(c_1, c_2, c_3)$ transforms S into T. The size of C is 3. Red (dotted) and green (solid) blocks indicate deletions and amplifications, respectively. CNT, CN transformation.

a CNO $c=(\ell, h, -1)$ is a *deletion* and $c=(\ell, h, 1)$ is an *amplification*. Given a CNP $V=(v_1, v_2, \ldots, v_n)$ and a CNO $c=(\ell, h, w)$, we define the operation $c(V)=(c(v_1), c(v_2), \ldots, c(v_n))$ as follows. For each $i \in \{1, 2, \ldots, n\}$, if $\ell \leq i \leq h$ and $v_i \geq 1$, then $c(v_i)=v_i+w$, otherwise (i.e., if $i<\ell$ or $i>h$ or $v_i=0$) $c(v_i)=v_i$. A triple $c=(\ell, h, w)$ with $h<\ell$ has no effect on the CNP, that is, $c(V)=V$. Given two CNPs, $S=(s_1, s_2, \ldots, s_n)$ (source) and $T=(t_1, t_2, \ldots, t_n)$ (target), a CN transformation (CNT) is a vector $C=(c_1, c_2, \ldots, c_m)$, where $m \in \mathbb{N}$ and each $c_i=(\ell_i, h_i, w_i)$ is a CNO, such that $C(S)=c_m(c_{m-1}(\cdots(c_1(S))))=T$. The *size* of C, denoted $|C|$, is m. An example is given in Figure 1. Finally, we denote the number of operations of weight $w \in \{-1, 1\}$ affecting s_i by $op(C, w, i)=|\{(\ell, h, w) \in C : \ell \leq i \leq h\}|$. For example, in Figure 1, $op(C, -1, 2)=1$.

The CN distance from S to T, dist(S, T), is the smallest size of a CNT C that satisfies $C(S)=T$, where if no such CNT exists, dist$(S, T)=\infty$. Note that *dist* is not symmetric. For example, for $S=(1)$ and $T=(0)$, dist$(S, T)=1$ but dist$(T, S)=\infty$. Given two CNPs, $S=(s_1, s_2, \ldots, s_n)$ and $T=(t_1, t_2, \ldots, t_n)$, the CNTP seeks dist$(S, T)$ (if one exists). We say that a CNT C is *optimal* if it realizes dist(S, T), that is, $|C|=$dist(S, T) (there may exist several optimal CNTs). We let $B=\max\{\max_{i=1}^{n}\{s_i\}, \max_{i=1}^{n}\{t_i\}\}$ denote the maximum CN in the input. Finally, for all $1 \leq i \leq n$, we define $u_i=s_i-t_i$.

3. AN ALGORITHM FOR CNTP

We first present an $O(nB^2)$-time and $O(B)$-space algorithm for CNTP, based on dynamic programming (Sections 3.1 and 3.2). Recall that B is the maximal integer in the input, so that algorithm is pseudopolynomial. Then, we modify this algorithm to run in linear time (Section 3.3). On a high level, the modification is based on the observation that the table used by the algorithm to store values of partial solutions can be described by $O(n)$ piecewise linear functions, where each function encapsulates $O(B)$ entries of the table. We show that each function has only three linear segments, and so, the computation of an entry can be performed in time $O(1)$ rather than $O(B)$. Furthermore, since each function can be represented in a compact manner, the size of table shrinks from $O(nB)$ to $O(n)$. The precise definitions of the table and the functions are given in Sections 3.2 and 3.3. Our proof of the correctness of the use of these functions requires a somewhat extensive case analysis that is presented separately in Section 3.4.

3.1. Key propositions

We start by developing `DpCntpAlg`, an $O(nB^2)$-time dynamic programming algorithm for CNTP. Let $(S=(s_1, s_2, \ldots, s_n),\ T=(t_1, t_2, \ldots, t_n))$ be the input. Observe that there exists a CNT C such that $C(S)=T$ if and only if there does not exist an index $1 \leq i \leq n$ such that $s_i=0$ and $t_i>0$. Since the existence of such an index can be determined in linear time (where, if such an index is found, we return ∞), we will assume that dist$(S, T)<\infty$. To simplify the presentation, we further assume w.l.o.g. that $t_1, t_n \neq 0$. Indeed, if $t_1=0$ or $t_n=0$, we can solve the input $(S'=(1, s_1, s_2, \ldots, s_n, 1), T'=(1, t_1, t_2, \ldots, t_n, 1))$ instead, since it holds that dist$(S, T)=$dist(S', T'). Finally, we assume w.l.o.g. for all $1 \leq i \leq n$, $s_i>0$. Indeed, if there exists $1 \leq i \leq n$ such that $s_i=0$, then also $t_i=0$, and we can solve the input $(S'=(s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n), T'=(t_1 \ldots, t_{i-1}, t_{i+1}, \ldots, t_n))$ since dist$(S, T)=$dist(S', T').

`DpCntpAlg` exploits four key observations about the nature of the problem at hand, summarized as follows: (1) it is sufficient to examine CNTs where all of the deletions precede all of the amplifications; (2)

it is sufficient to examine CNTs that do not contain both a deletion that affects s_i but not s_{i+1} and a deletion that affects s_{i+1} but not s_i, and the same is true for amplifications; (3) when seeking an optimal solution, it is not necessary to store information indicating how many deletions/amplifications affect s_i if $t_i = 0$; and (4) the maximum number of deletions/amplifications that affect each s_i can be bounded by B.

To formally state the first observation, we need the following definition.

Definition 1. *A CNT $C = (c_1, c_2, \ldots, c_m)$ is ordered if for all $1 \leq i < j \leq m$, if c_j is a deletion, then c_i is also a deletion.*

Proposition 1. *There exists an optimal ordered CNT.*

We note that the ''opposite'' proposition, stating that there exists an optimal CNT where all of the amplifications precede all of the deletions, does not hold: consider, for example, $S = (1, 1, 1, 1, 1)$ and $T = (2, 0, 2, 0, 2)$. To prove this proposition, we will need the following claim.

Claim 1. *Let $C = (c_1, c_2, \ldots, c_m)$ be an optimal CNT and let i be an index such that $c_i = (\ell_i, h_i, 1)$ and $c_{i+1} = (\ell_{i+1}, h_{i+1}, -1)$. Then, there exists an optimal CNT $C' = (c_1, \ldots, c_{i-1}, c'_i, c'_{i+1}, c_{i+2}, \ldots, c_m)$, where $c'_i = (\ell'_i, h'_i, w'_i)$ and $c'_{i+1} = (\ell'_{i+1}, h'_{i+1}, w'_{i+1})$, such that one of the following conditions holds.*

1. $(h'_i - \ell'_i) + (h'_{i+1} - \ell'_{i+1}) < (h_i - \ell_i) + (h_{i+1} - \ell_{i+1})$.
2. $(h'_i - \ell'_i) + (h'_{i+1} - \ell'_{i+1}) = (h_i - \ell_i) + (h_{i+1} - \ell_{i+1})$ *and* $w'_i = -1$.

Proof. Consider the following exhaustive case analysis (Fig. 2). ■

I. $h_i < \ell_{i+1}$ or $h_{i+1} < \ell_i$: In this case, the segments corresponding to c_i and c_{i+1} are disjoint. Thus, we can simply define $c'_i = c_{i+1}$ and $c'_{i+1} = c_i$. Then, Condition 2 is satisfied.

II. $\ell_i \leq \ell_{i+1} \leq h_i \leq h_{i+1}$: Define $c'_i = (h_i + 1, h_{i+1}, -1)$ and $c'_{i+1} = (\ell_i, \ell_{i+1} - 1, 1)$. For any CNP $V = (v_1, v_2, \ldots, v_n)$, $c'_{i+1}(c'_i(V)) = c_{i+1}(c_i(V))$. This argument holds because an application of c_i, followed by an application of c_{i+1}, does not change any entry v_k such that $\ell_{i+1} \leq k \leq h_i$. We have that $C'(S) = T$. Since $|C'| = |C|$, C' is an optimal CNT. Now, Condition 1 is satisfied.

III. $\ell_{i+1} \leq \ell_i \leq h_{i+1} \leq h_i$: Define $c'_i = (\ell_{i+1}, \ell_i - 1, -1)$ and $c'_{i+1} = (h_{i+1} + 1, h_i, 1)$. As in the second case, we obtain an optimal CNT that satisfies Condition 1.

IV. $\ell_i \leq \ell_{i+1} \leq h_{i+1} \leq h_i$: Define $c'_i = (\ell_i, \ell_{i+1} - 1, 1)$ and $c'_{i+1} = (h_{i+1} + 1, h_i, 1)$. As in the second case, we obtain an optimal CNT that satisfies Condition 1.

V. $\ell_{i+1} \leq \ell_i \leq h_i \leq h_{i+1}$: Define $c'_i = (\ell_{i+1}, \ell_i - 1, -1)$ and $c'_{i+1} = (h_i + 1, h_{i+1}, -1)$. As in the second case, we obtain an optimal CNT that satisfies Condition 1.

As we show below, Claim 1 implies the existence of an ordered optimal CNT. In each of the cases in Claim 1, a local change is made in the CNT. Note, however, that just performing enough local operations does not guarantee reaching an ordered optimal CNT. For example, in a CNT with three consecutive CNOs, $c_i = (\ell_i, h_i, 1)$, $c_{i+1} = (\ell_{i+1}, h_{i+1}, 1)$, $c_{i+2} = (\ell_{i+2}, h_{i+2}, -1)$, one may loop between changing c_{i+1} into a deletion and then into an amplification.

Proof (Proof of Proposition 1). Let $\mathcal{C}$ be the set of optimal CNTs, and suppose, by way of contradiction, that it does not contain an ordered CNT. The three following phases sieve some solutions out of $\mathcal{C}$. Informally, we initially consider only optimal CNTs that minimize the sum of the sizes of the segments corresponding to their CNOs ($\mathcal{C}^1$); then, we further consider only the CNTs whose first amplification is as late as possible ($\mathcal{C}^2$); finally, we only take the CNTs whose first deletion after their first amplification is as early as possible ($\mathcal{C}^3$). An illustration is given in Figure 3.

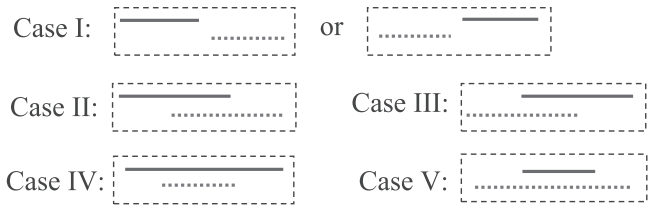


FIG. 2. The proof of Claim 1. The green (solid) lines correspond to c_i, and the red (dotted) lines correspond to c_{i+1}.

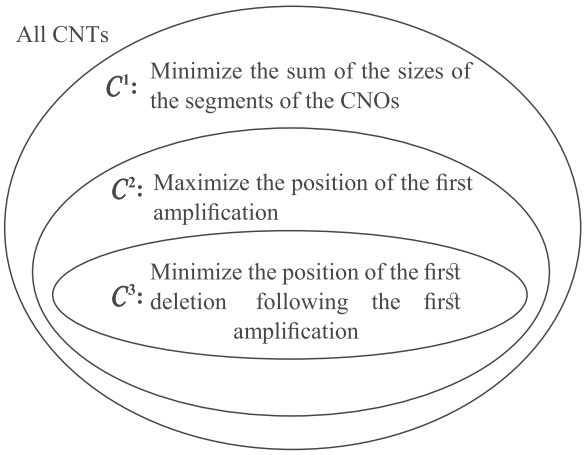


FIG. 3. The proof of Proposition 1.

- Given $C=(c_1, c_2, \ldots, c_m) \in \mathcal{C}$, define $x(C)=\sum_{i=1}^{m}(h_i-\ell_i)$. Let $\mathcal{C}^1$ be the set of every $C \in \mathcal{C}$ for which there does not exist $C' \in \mathcal{C}$ such that $x(C) > x(C')$.
- Given $C=(c_1, c_2, \ldots, c_m) \in \mathcal{C}^1$, let $y(C)$ be the largest index $0 \le i \le m$ such that for all $1 \le j \le i$, c_j is a deletion. Note that $y(C)=0$ if and only if c_1 is an amplification. Let $\mathcal{C}^2$ be the set of every $C \in \mathcal{C}^1$ for which there does not exist $C' \in \mathcal{C}^1$ such that $y(C) < y(C')$.
- Given $C=(c_1, c_2, \ldots, c_m) \in \mathcal{C}^2$, let $z(C)$ be the smallest index $i \in \{y(C)+1, \ldots, m\}$ such that c_i is a deletion. By the definition of $y(C)$ and since C is not ordered, we have that $z(C)$ is well defined and $z(C) \ge y(C)+2$. Let $\mathcal{C}^3$ be the set of every $C \in \mathcal{C}^2$ for which there does not exist $C' \in \mathcal{C}^2$ such that $z(C) > z(C')$.

Since $\mathcal{C} \neq \emptyset$, we have that $\mathcal{C}^3 \neq \emptyset$. Thus, we can let $C=(c_1, c_2, \ldots, c_m)$ be a solution in $\mathcal{C}^3$. Let i be the smallest index such that c_i is an amplification and c_{i+1} is a deletion. Now, consider the conditions in Claim 1: if Condition 1 holds, we have a contradiction to the fact that $C \in \mathcal{C}^1$, while if Condition 2 holds, we have a contradiction either to the fact that $C \in \mathcal{C}^2$ (if $i=1$ or c_{i-1} is a deletion) or to the fact that $C \in \mathcal{C}^3$ (otherwise). Thus, we conclude that $\mathcal{C}$ contains an ordered CNT.

Definition 2. *A CNT C is elongated if for all $1 \le i < n$ and $w \in \{-1, 1\}$,*

$$\min\{op(C, w, i), op(C, w, i+1)\} = |\{(\ell, h, w) \in C : \ell \le i, i+1 \le h\}|.$$

Equivalently, C is elongated if no two amplifications (or deletions) ''dovetail,'' that is, one ending at i and the other starting at $i+1$. It is clear that for any CNT C, the inequality $\ge$ holds above (since $\{(\ell, h, w) \in C : \ell \le i, i+1 \le h\}$ is a subset of both $\{(\ell, h, w) \in C : \ell \le i \le h\}$ and $\{(\ell, h, w) \in C : \ell \le i+1 \le h\}$). Our second key proposition implies the inequality $\le$ holds as well. An example for an elongated CNT is given in Figure 4A.

To prove Proposition 2, we will need the following claim.

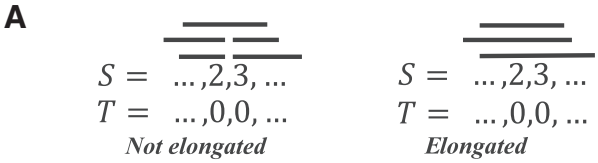
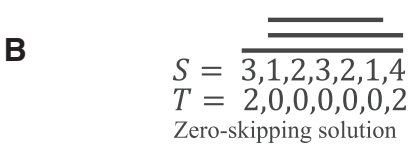


FIG. 4. **(A)** Elongated and nonelongated CNTs. **(B)** A zero-skipping solution. The top lines indicate the range of deletions.



Claim 2. *Let $C=(c_1, c_2, \ldots, c_m)$ be an optimal ordered CNT, and let $1 \leq i < j \leq m$ be indices such that either both c_i and c_j are deletions or both c_i and c_j are amplifications. Then, the CNT C' obtained from C by swapping the locations of c_i and c_j is also an optimal ordered CNT.*

Proof. Clearly, C' is ordered and $|C'|=|C|$. Thus, it is sufficient to show that $C'(S)=C(S)$. Observe that because C is ordered, for any $1 \leq q \leq n$, the value of the q^{st} CN in $C(S)$ is $x+y$, where $x=\max\{s_q - op(C, -1, q), 0\}$, $y=0$ if $x=0$, and $y=op(C, 1, q)$ otherwise. By the definition of C' (which is also ordered and contains the same CNOs as C), this is also the value of the q^{st} CN in $C'(S)$. ■

We are now ready to show the following property.

Proposition 2. *Every ordered optimal CNT is elongated.*

Proof. Let $C=(c_1, c_2, \ldots, c_m)$ be an optimal ordered CNT. Suppose that, by way of contradiction, C is not elongated. Thus, there exist $1 \leq i < n$ and $w \in \{-1, 1\}$ such that

$$\min\{op(C, w, i), op(C, w, i+1)\} > |\{(\ell, h, w) \in C : \ell \leq i, i+1 \leq h\}|.$$

Therefore, C contains two CNOs $c_p=(\ell_p, h_p, w)$ and $c_q=(\ell_q, h_q, w)$ such that $h_p=i$ and $\ell_q=i+1$. By Claim 2, we can assume that $p=q+1$. Now, by removing c_p and replacing c_q by the CNO $c=(\ell_p, h_q, w)$, we obtain a CNT C' such that $C'(S)=T$. However, $|C'| < |C|$, which contradicts the optimality of C. ■

To formalize our third key proposition, we need the following definition.

Definition 3. *A CNT C is zero-skipping if for every $1 \leq i < j \leq n$ such that for all $i < r \leq j, t_r=0$ we have*

$$op(C, -1, j) = \max\left\{\max_{r=i+1}^{j}\{s_r\}, op(C, -1, i)\right\}, \text{ and } op(C, 1, j) = op(C, 1, i).$$

In words, for a block of consecutive zeros in the target profile, all deletions that span the block also include its flanking positions. An example of a zero-skipping CNT is given in Figure 4B.

Proposition 3. *There exists an optimal ordered zero-skipping CNT.*

Proof. By Proposition 1, there is an optimal ordered CNT $C=(c_1, c_2, \ldots, c_m)$. If C is zero-skipping, we are done, and thus we next suppose that it does not. Thus, there exists $1 \leq i < j \leq n$ such that $t_r=0$ for all $i < r \leq j$, for which at least one of the following conditions is satisfied. ■

1. $op(C, -1, j) \neq \max\{\max_{r=i+1}^{j}\{s_r\}, op(C, -1, i)\}$.
2. $op(C, 1, j) \neq op(C, 1, i)$.

We can assume w.l.o.g. that j is the smallest index that is larger than i for which at least one of the above conditions is satisfied. Thus, at least one of the following conditions is satisfied.

1. $op(C, -1, j) \neq \max\{s_j, op(C, -1, j-1)\}$.
2. $op(C, 1, j) \neq op(C, 1, j-1)$.

Since $t_j=0$, $|\{(\ell, h, -1) \in C : \ell \leq j \leq h\}| \geq s_j$. Moreover, because C is ordered and $t_j=0$, we can replace each CNO $c=(\ell, h, w)$ in C such that $h=j-1$ by the CNO $c'=(\ell, j, w)$. Thus, we overall obtain an optimal ordered CNT C', such that, if it is not zero-skipping (in which case we are done), at least one of the following conditions is satisfied.

1. $op(C, -1, j) > \max\{s_j, op(C, -1, j-1)\}$.
2. $op(C, 1, j) > op(C, 1, j-1)$.

Since C' is ordered and $t_j=0$, we can choose a CNO $c=(\ell, h, w)$ in C' such that $\ell=j$, as well as $w=-1$ if the first condition is satisfied and $w=1$ otherwise, and replace it by the CNO $c'=(j+1, h, w)$. This operation results in an optimal ordered CNT. By repeating it enough times, we obtain an optimal ordered CNT that is zero-skipping. ■

For a position with positive target value, knowing the number of deletions that affected it uniquely determines the number of amplifications that affected it. This simple fact will help the efficiency of our procedures. Formally:

Observation 1. *Let $1 \leq i \leq n$ be an index such that $t_i > 0$, and let $C = (c_1, c_2, \ldots, c_m)$ be a CNT such that $C(S) = T$. Then, $op(C, 1, i) = -u_i + op(C, -1, i)$.*

Finally, we formalize our fourth key proposition.

Definition 4. *A CNT C is bounded if for all $1 \leq i \leq n$ and every $w \in \{-1, 1\}$, we have $op(C, w, i) \leq B$.*

Proposition 4. *Every optimal ordered CNT that is zero-skipping is also bounded.*

Proof. Let C be an optimal ordered CNT that is zero-skipping. Suppose, by way of contradiction, that C is not bounded. That is, there exists $1 \leq i \leq n$ and $w \in \{-1, 1\}$ such that $op(C, w, i) > B$. First suppose that $t_i > 0$. Then, since C is ordered and $C(S) = T$, we have that $w = 1$. However, this contradicts the correctness of Observation 1. Thus, we can next suppose that $t_i = 0$, which also implies that $i > 1$. We also assume w.l.o.g. that i is the smallest index such that $op(C, w, i) > B$. Therefore, at least one of the following conditions is satisfied.

1. $op(C, -1, j) > \max\{s_j, op(C, -1, j-1)\}$.
2. $op(C, 1, j) > op(C, 1, j-1)$.

Thus, we necessarily obtain a contradiction to the fact that C is zero-skipping. ■

3.2. An $O(nB^2)$-time algorithm for CNTP

On a high-level, the dynamic programming algorithm works as follows. It considers increasing prefixes $S^i = (s_1, s_2, \ldots, s_i)$ and $T^i = (t_1, t_2, \ldots, t_i)$ of the input. It computes a table M having $n(B+1)$ entries where $M[i, d]$ is the best value of a solution on (S^i, T^i) that uses exactly d deletions that affect the i^{th} position. The parameter d ranges between zero and B, and the values for each i are computed based on values $M[j, \cdot]$ for a single specific $j < i$. In particular, at each point of time, only two rows of the table M are stored. By Propositions 1–4, the algorithm considers only ordered, elongated, zero-skipping and bounded solutions. We call such solutions *good*.

More formally, given $1 \leq i \leq n$ and $0 \leq d \leq B$, we say that a CNT C is an (i, d)-*CNT* if $C(S^i) = T^i$, $d = op(C, -1, i)$, and C is good. We say that an (i, d)-CNT C is *optimal* if there is no (i, d)-CNT C' such that $|C'| < |C|$. Our goal will be to ensure that each entry $M[i, d]$ stores the size of an optimal (i, d)-CNT, where if no such CNT exists, it stores ∞. We do not compute entries $M[i, d]$ such that $t_i = 0$; indeed, by relying on Property 3, we are able to skip such entries (although our recursive formula does consider CNs s_i referring to indices i such that $t_i = 0$). In this context, observe that any ordered CNT C such that $C(S) = T$ consists of at least u_i deletions that affect s_i, and if $t_i > 0$, it cannot consist of more than $s_i - 1$ such deletions (since after decreasing s_i to 0, it remains 0). Moreover, if $u_i \leq d < s_i$, there exists an (i, d)-CNT—by independently adjusting the value of each position $< i$ to its target position and the value at position i with d deletions, using operations of span 1.

Observation 2. *Given $1 \leq i \leq n$ such that $t_i > 0$ and $0 \leq d \leq B$, there exists an (i, d)-CNT if and only if $u_i \leq d < s_i$.*

In case $s_i < t_i$, Observation 2 states that there exists an (i, d)-CNT if and only if $d < s_i$. In light of this observation, we will use the following assumption.

Assumption 1. *In the computation below, we assume that $\max\{u_i, 0\} \leq d < s_i$. Entries $M[i, d]$ for which it is not true that $\max\{u_i, 0\} \leq d < s_i$ store ∞.*

By Observation 1, if a solution involved d deletions at position i with $t_i > 0$, then it involved $-u_i + d$ amplifications at that position. For convenience denote that number by $a(i, d) = -u_i + d$ for all $1 \leq i \leq n$ satisfying $t_i > 0$ and $\max\{u_i, 0\} \leq d < s_i$, and $a(i, d) = \infty$ otherwise.

For input profiles S, T, the algorithm precomputes two vectors. Given an index $1 < i \leq n$ such that $t_i > 0$, let prev(i) denote the largest index $j < i$ such that $t_j > 0$. Moreover, if prev$(i) = i - 1$, let $Q_i = 0$, and otherwise let $Q_i = \max_{\text{prev}(i) < j < i}\{s_j\}$. A zero-skipping solution (Fig. 5) will skip the positions between i and prev(i) in the computation, but will make sure to perform at least Q_i deletions spanning the skipped positions.

Initialization: The initialization step sets all entries $M[1, d]$ as follows.

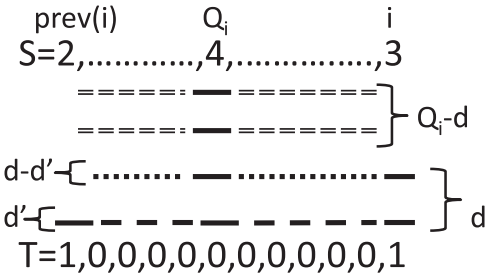


FIG. 5. Zero-skipping in the recursive formula. T has a maximal block of zeros between positions prev(i) and i, S has values 2 and 3, respectively, in these positions and a maximum value 4 within the interval of genes, attained at position Q_i. d' deletions can be elongated from prev(i) up to position i. $d-d'$ deletions can be extended forward up to position i and backward to position prev(i)+1. In addition, Q_i-d additional deletions are needed to delete Q_i.

$$\mathrm{M}[1, d] \leftarrow d + a(1, d).$$

Recursion: If $t_i = 0$ position i is skipped. Suppose that $i > 1$, $t_i > 0$, and $\max\{u_i, 0\} \le d < s_i$. The order of the computation is determined by the first argument. The computation is summarized in the following formula and illustrated in Figure 5.

$$\begin{aligned}\mathrm{M}[i, d] \leftarrow \min_{0 \le d' \le B} \{&\mathrm{M}[\mathrm{prev}(i), d'] + \max\{d - d', 0\} \\ &+ \max\{a(i, d) - a(\mathrm{prev}(i), d'), 0\} + \max\{Q_i - \max\{d, d'\}, 0\}\}\end{aligned} \tag{1}$$

Roughly speaking, to compute $M[i, d]$ we look back to the previous nonzero position in T, and for each value d' in that position add the difference from d if needed, the number of amplifications to be added if needed, and the number of additional deletions if such are needed to take care of the zero positions that were skipped. After filling the table M, `DpCntpAlg` returns $\min_{0 \le d \le B}$ M[n, d]. The full algorithm is given in Algorithm 1. An example of a partially filled table is given in Figure 6.

Algorithm 1: `DpCntpAlg`

Input: S, T, Q, *prev*
Output: *dist*(S, T)
  for $d = 1, \ldots, B$ **do**
    $M[1, d] \leftarrow d + a(1, d)$
  end for
  for $i = 2, \ldots, n$, $t_i > 0$ **do**
    for $d = 0, \ldots, B$ **do**
      if $\max\{u_i, 0\} \le d < s_i$ **then**
        $\mathrm{M}[i,d] \leftarrow \min_{0 \le d' \le B}\{\mathrm{M}[\mathrm{prev}(i), d'] + \max\{d - d', 0\} + \max\{a(i, d) - a(\mathrm{prev}(i), d'), 0\}$
      else
        $\mathrm{M}[i, d] \leftarrow \infty$
      end if
    end for
  end for
  return $\min_{0 \le d \le B}$ M[n, d]

$M[i, d]$

	i = 1	i = 7
d = 0	∞	∞
d = 1	1	∞
d = 2	2	3
d = 3	∞	4
d = 4	∞	∞

FIG. 6. The DP $M[i, d]$ matrix for the two CNPs in Figure 4B.

Correctness: First, we claim that the entries of the table M are computed properly.

Lemma 1. *For all $1 \leq i \leq n$ such that $t_i > 0$ and for all $0 \leq d \leq B$, $M[i, d]$ stores the size of an optimal (i, d)-CNT, where if no such CNT exists, it stores ∞.*

Proof. We prove the lemma by induction on the order of the computation. ■

The correctness of the initialization step follows from the definition of an (i, d)-CNT and Observation 1.

Now, fix $1 < i \leq n$ such that $t_i > 0$, and fix $\max\{u_i, 0\} \leq d < s_i$. Let m be the size of an optimal (i, d)-CNT. Suppose that the lemma is correct for all $i' < i$ and $0 \leq d' \leq B$. We need to show that $\mathrm{M}[i, d] = m$.

First Direction: First, we show that $\mathrm{M}[i, d] \leq m$. Let $C = (c_1, c_2, \ldots, c_m)$ be an optimal (i, d)-CNT, and for all $1 \leq j \leq m$, denote $c_j = (\ell_j, h_j, w_j)$. For all $1 \leq j \leq m$, let $c'_j = (\ell_j, \min\{h_j, \mathrm{prev}(i)\}, w_j)$. Now, define $C' = (c'_1, c'_2, \ldots, c'_m)$. We further let $\hat{C} = (\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_q)$ denote the CNT obtained from C' by removing all of the CNOs $c = (\ell, h, w)$ such that $h < \ell$. Denote $\hat{d} = op(\hat{C}, -1, \mathrm{prev}(i))$. Observe that $\hat{d} \leq B$ and that $\hat{C}$ is a $(\mathrm{prev}(i), \hat{d})$-CNT (because C is an (i, d)-CNT). Therefore, by the induction hypothesis, $\mathrm{M}[\mathrm{prev}(i), \hat{d}] \leq q$ (recall that $q = |\hat{C}|$). If $\mathrm{prev}(i) = i - 1$, then $Q_i = 0$ and since C is ordered and elongated, by Observation 1 we have that $m - q = \max\{d - \hat{d}, 0\} + \max\{a(i, d) - a(prev(i), \hat{d}), 0\}$. Thus, by the recursive formula, in this case we get that $\mathrm{M}[i, d] \leq m$.

Now, suppose that $\mathrm{prev}(i) < i - 1$. Then, since C is ordered and zero-skipping, and by the definition of Q_i, the two following conditions hold.

1. $op(C, -1, i-1) = \max\{Q_i, op(C, -1, \mathrm{prev}(i))\}$.
2. $op(C, 1, i-1) = op(C, 1, \mathrm{prev}(i))$.

Thus, since C is ordered and elongated, by Observation 1 we have that $m - q = \max\{d - \hat{d}, 0\} + \max\{a(i, d) - a(\mathrm{prev}(i), \hat{d}), 0\} + \max\{Q_i - \max\{d, \hat{d}\}, 0\}$. Again, by the recursive formula, this implies that $\mathrm{M}[i, d] \leq m$.

Second Direction: Next, we show that $\mathrm{M}[i, d] \geq m$. To this end, it is sufficient to show that there exists an (i, d)-CNT C such that $\mathrm{M}[i, d] \geq |C|$. Let $\hat{d}$ be an argument d' at which the value computed by using the recursive formula is minimized. By the inductive hypothesis, there exists a $(\mathrm{prev}(i), \hat{d})$-CNT $\hat{C} = (\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_q)$ such that $\mathrm{M}[\mathrm{prev}(i), \hat{d}] \geq q$. For all $1 \leq j \leq q$, denote $\hat{c}_j = (\ell_j, h_j, w_j)$. Now, if $\mathrm{prev}(i) = i - 1$, define $\widetilde{C} = \hat{C}$, else define $\widetilde{C}$ as follows. For all $1 \leq j \leq q$, let $\tilde{c}_j = (\ell_j, \widetilde{h}, w_j)$, where $\widetilde{h} = h_j$ if $h_j < \mathrm{prev}(i)$ and $\widetilde{h} = i - 1$ otherwise. Let $\widetilde{C} = (\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_q)$. Moreover, as long as there exists $\mathrm{prev}(i) < j < i$ such that $op(\widetilde{C}, -1, j) < s_j$, choose the smallest such j, and append to the beginning of $\widetilde{C}$ the CNO $(j, i-1, -1)$. Let C' be the CNT obtained at the end of this process. Denote $C' = (c'_1, c'_2, \ldots, c'_r)$, and for all $1 \leq j \leq r$, denote $c'_j = (\ell'_j, h'_j, w'_j)$. Now, let p and q be the number of deletions and amplifications in C' whose segments include $i - 1$, respectively. If $p < d$, append to the beginning of C' $d - p$ "dummy" deletions of the form $(i, i-1, -1)$, and if $a(i, d) < q$, append to the end of C' $a(i, d) - q$ "dummy" amplifications of the form $(i, i-1, 1)$. Let $C'' = (c''_1, c''_2, \ldots, c''_k)$ be the resulting CNT, and for all $1 \leq j \leq k$, denote $c''_j = (\ell''_j, h''_j, w''_j)$. Finally, we define C as follows. Let D (A) be a set of exactly d deletions (resp. amplifications) in C'' whose second argument is $i - 1$. We let C be defined as C'', except that each CNO $(\ell, h, w) \in D \cup A$ is replaced by the CNO (ℓ, i, w). It is straightforward to verify that C is an (i, d)-CNT such that $|C| = q + \max\{d - \hat{d}, 0\} + \max\{a(i, d) - a(\mathrm{prev}(i), \hat{d}), 0\} + \max\{Q_i - \max\{d, \hat{d}\}, 0\}$, which concludes the correctness of the second direction.

Now, we turn to consider the correctness and running time of `DpCntpAlg`.

Theorem 1. `DpCntpAlg` *solves* CNTP *in time $O(nB^2)$ and space $O(B)$.*

Proof. The table M contains $O(nB)$ entries, and each entry can be computed in time $O(B)$. Therefore, the time complexity of `DpCntpAlg` is bounded by $O(nB^2)$. Moreover, for the computation of $\mathrm{M}[i, \cdot\,]$, it is only necessary to keep $O(B)$ entries for position $\mathrm{prev}(i)$, and therefore, the space complexity is bounded by $O(B)$. Since every (n, d)-CNT C satisfies $C(S) = T$, and since for every good optimal CNT C, there exists $0 \leq d \leq B$ such that C is an (n, d)-CNT, we have that Lemma 1 implies that `DpCntpAlg` returns the smallest size of a good optimal CNT (if such a CNT exists). By Propositions 1–4, such a CNT indeed exists, and therefore `DpCntpAlg` solves CNTP. ■

3.3. A linear-time algorithm for CNTP

In this section we show how to modify `DpCntpAlg` to obtain an algorithm, called LinearCntpAlg, that solves CNTP in linear time. The central lemma that leads to this improvement states that each column in the table M can be described by a piecewise linear function of at most three segments.

To present this lemma, we need the following notation. For all $i \in \{1, 2, \ldots, n\}$ such that $t_i > 0$, let $d_i^{min} = \max\{u_i, 0\}$ and $d_i^{max} = \max\{s_i - 1, 0\}$ be the least and largest values of d for which $M[i, d]$ is finite. Now, the function $f_i : \{d_i^{min}, \ldots, d_i^{max}\} \to \mathbb{N} \cup \{0\}$ will satisfy $f_i(d) = \mathrm{M}[i, d]$. Observe that the function f_i is discrete. We stress that in this section, we do not explicitly compute the entries of M—the definition of the functions concerns the values that would have been stored in these entries if they were computed by using `DpCntpAlg`.

Lemma 2. *For each $i \in \{1, 2, \ldots, n\}$ such that $t_i > 0$, there exist $base_i, a_i, b_i \in \mathbb{N} \cup \{0\}$ such that for all $d \in \{d_i^{min}, \ldots, d_i^{max}\}$:*

$$f_i(d) = \begin{cases} base_i & \text{if} \quad d_i^{min} \le d \le a_i \\ (base_i - a_i) + d & \text{if} \quad a_i \le d \le b_i \\ (base_i - a_i - b_i) + 2d & \text{if} \quad b_i \le d \le d_i^{max} \end{cases}$$

Moreover, $base_1, a_1$ and b_1 can be computed in constant time, and for each $i \in \{2, 3, \ldots, n\}$ such that $t_i > 0$, given $base_{\mathrm{prev}(i)}, a_{\mathrm{prev}(i)}$ and $b_{\mathrm{prev}(i)}$, $base_i, a_i$ and b_i can be computed in constant time.

An example is given in Figure 7. The proof is based on Lemma 1 and on an exhaustive case analysis, which, for the sake of clarity of presentation, is handled separately in Section 3.4.

Our algorithm, `LinearCntpAlg`, performs the following computation, using `PiecewiseAlg`, an algorithm that computes $base_i$, a_i, and b_i in constant time. That algorithm is described in the next subsection.

We are now ready to prove our main result.

Algorithm 2: `LinearCntpAlg`

Input: S, T, Q, *prev*
Output: *dist*(S, T)
  $base_0 \leftarrow 0$; $a_0 \leftarrow 0$; $b_0 \leftarrow 0$.
  for $i = 1, \ldots, n$, $t_i > 0$ **do**
    $base_i$; a_i, $b_i \leftarrow$ `PiecewiseAlg`(s_i, t_i, Q_i, $base_{prev(i)}$, $a_{prev(i)}$, $b_{prev(i)}$).
  end for
  return $base_n$

Theorem 2. `LinearCntpAlg` *solves* CNTP *in time $O(n)$ and space $O(1)$.*

Proof. According to Lemma 2, $f_i(d) = \mathrm{M}[i, d]$ is a piecewise linear function described by three values: $base_i$, a_i and b_i. Lemma 2 shows that `PiecewiseAlg` calculates these values in constant time and space given the previous values. The time and space complexity of `LinearCntpAlg` follow directly. ■

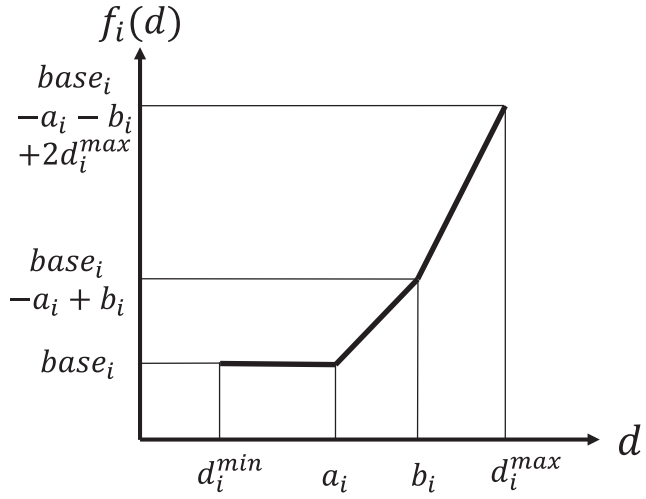


FIG. 7. An example of the piecewise linear function $f_i(d)$ described in Lemma 2. The number of segments is three but can be smaller, depending on the values involved.

Now, by the correctness of DpCntpAlg, it is sufficient to prove that LinearCntpAlg returns the value $\min_{0\le d\le B} \mathrm{M}[n,d]$. By Observation 2, $\min_{0\le d\le B} \mathrm{M}[n,d] = \min_{d_n^{min}\le d\le d_n^{max}} \mathrm{M}[n,d]$. By Lemma 2, we further have that $\min_{d_n^{min}\le d\le d_n^{max}} \mathrm{M}[n,d] = base_n$. Thus, by the inductive proof of Lemma 2, we conclude that LinearCntpAlg solves CNTP. ■

3.4. Case analysis

This section is to prove the correctness of Lemma 2. That is, we want to show that $f_i(d)$ is a piecewise linear function described by three parameters, and these parameters can be calculated in constant time. To this end, let $j = \mathrm{prev}(i)$ and $R_i = u_j - u_i$. Accordingly, the term $a(i,d) - a(j,d')$ can be written as $R_i + d - d'$. Moreover, let d'_{opt} be the argument d' that minimizes the recursive formula we use to compute $\mathrm{M}[i,d]$ under certain conditions that will be clear from context.

We prove Lemma 2 by induction on i. To simplify the proof, let $a_0 = b_0 = base_0 = 0$ and $f_0(d) = 2d$ for every $0 \le d \le B$. This definition is equivalent to adding the new entries $s_0 = t_0 = B+1$ (which do not affect the distance from S to T), and thus, it can serve as the basis of our induction. Next, suppose that Lemma 2 holds for $j = \mathrm{prev}(i) < i$, we will prove that it holds for i.

The proof is based on an exhaustive case analysis that examines the position of Q_i relative to d_j^{min}, a_j, b_j, and d_j^{max}, as well as the sign of R_i. For example, Case 2(a)ii is defined by the conditions $d_j^{min} \le Q_i \le a_j$, $R_i \ge 0$, and $a_j - R_i \le Q_i$. In each case, we analyze the behavior of $\mathrm{M}[i,d]$ as we increase d. More precisely, we examine several intervals that together contain all of the values that can be assigned to d. For example, in the abovementioned case, we consider the intervals $d \le a_j - R_j$, $a_j - R_j \le d \le Q_i$, and $Q_i \le d$. For each interval, we let d'_{opt} be an argument d' that minimizes $\mathrm{M}[i,d]$ under the conditions of the examined case. These conditions along with d'_{opt} allow us to remove the minimization and maximization functions from the formula defining $\mathrm{M}[i,d]$, and thus, we obtain $f_i(d)$. In the latter example, if $d \le a_j - R_j$ we can choose $d'_{opt} = a_j$ and get $f_i(d) = \mathrm{M}[i,d] = \mathrm{M}[j,a_j] + \max\{d - a_j, 0\} + \max\{R_i + d - a_j, 0\} + \max\{Q_i - \max\{d, a_j\}, 0\}\} = base_j$. As a corollary of the analysis, we get that indeed $f_i(d)$ is piecewise linear, and that a_i, b_i, and $base_i$ can be calculated in constant time given a_j, b_j, $base_j$, R_i, and Q_i.

The full case analysis is given in the Appendix. The analysis shows that in all cases, $f_i(d)$ is indeed a piecewise linear function with at most three linear segments defined by some a_i, b_i, and $base_i$. After applying straightforward operations that reorganize the analysis (to present the results in a compact manner), we obtain the algorithm PiecewiseAlg, whose pseudocode is given below. This algorithm performs the iterative step of LinearCntpAlg, that is, it calculates a_i, b_i, $base_i$ given a_j, b_j, $base_j$, and Q_i in constant time and space.

PiecewiseAlg first calculates R_i, d_i^{min} and d_i^{max} based on s_i and t_i. Next, according to the sign of R_i and the relative position of Q_i in comparison to the previous a_j and b_j, the algorithm calculates the structure of $f_i(d)$ defined by a_i and b_i. Finally, since $f_i(d)$ is defined only for the range $d_i^{min} \le d \le d_i^{max}$, we calculate $base_i = f_i(d_i^{min})$. Similarly, we limit the values of a_i and b_i to that range.

Algorithm 3: PiecewiseAlg

Input: $s_i, t_i, Q_i, a_j, b_j, base_j$
Output: $a_i, b_i, base_i$

$R_i \leftarrow u_j - u_i$
$d_i^{min} \leftarrow \max\{u_i, 0\}$
$d_i^{max} \leftarrow \max\{s_i - 1, 0\}$
$a_i \leftarrow \min\{\max\{a_j, Q_i\}, b_j - \min\{R_i, 0\}\} - \max\{R_i, 0\}$
$b_i \leftarrow \max\{Q_i, b_j - \min\{R_i, 0\}\}$

$$base_i \leftarrow base_j + \max\{Q_i - a_j, 0\} + \begin{cases} 0 & \text{if} \quad d_i^{min} \le a_i \\ d_i^{min} - a_i & \text{if} \quad a_i < d_i^{min} \le b_i \\ 2d_i^{min} - a_i - b_i & \text{if} \quad b_i < d_i^{min} \le d_i^{max} \end{cases}$$

$a_i \leftarrow \max\{d_i^{min}, \min\{a_i, d_i^{max}\}\}$; $b_i \leftarrow \max\{a_i, \min\{b_i, d_i^{max}\}\}$
return $base_i, a_i, b_i$

4. CONCLUSION

In this article, we introduced the study of distances between CNPs from a theoretical point of view. We focused on one fundamental problem, CNTP, and showed that it is solvable in linear time and constant space. To this end, we proved several properties of CNTP that may be useful in solving other problems involving CNPs. Our algorithm can be modified to return a transformation that realizes $dist(S, T)$ in linear time and linear space by backtracking the dynamic programming vector. We have implemented the algorithm as well as a linear programming formulation of CNTP, and the implementations are available on request.

Many computational and combinatorial aspects in the analysis of distances between CNPs require further research. Indeed, this article can be viewed as a first step toward understanding them. In our follow-up article by El-Kebir et al. (2016), we investigated a generalization of CNTP where the input is a set of profiles, and one seeks to construct a tree with the profile labels at the leaves and additional profile labeling of internal nodes that minimizes the transformation distances along the edges. We showed this problem is NP-hard and gave an Integer Linear Programming (ILP) formulation to solve it. Additional directions for further research involve the introduction of edit operations other than basic segmental deletions and amplifications, dealing with phasing of the profiles, as well as the handling of noise.

ACKNOWLEDGMENTS

We thank the referees for many helpful comments. This study was supported by the Israeli Science Foundation (grant 317/13), the Israel Cancer Association, and the Dotan Hemato-Oncology Research Center at Tel Aviv University. R.Z. was supported by fellowships from the Edmond J. Safra Center for Bioinformatics at Tel Aviv University and from the Israeli Center of Research Excellence (I-CORE) Gene Regulation in Complex Human Disease (Center No 41/11). M.Z. was supported by a fellowship from the I-CORE in Algorithms and the Simons Institute for the Theory of Computing in Berkeley and by the Postdoctoral Fellowship for Women of Israel's Council for Higher Education.

AUTHOR DISCLOSURE STATEMENT

No competing financial interests exist.

REFERENCES

Abo, R.P., Ducar, M., Garcia, E.P., et al. 2015. BreaKmer: Detection of structural variation in targeted massively parallel sequencing data using kmers. *Nucleic Acids Res.* 18; 43(3):e19.

Chowdhury, S.A., Gertz, E.M., Wangsa, D., et al. 2015. Inferring models of multiscale copy number evolution for single-tumor phylogenetics. *Bioinformatics* 31, i258–i267.

Chowdhury, S.A., Shackney, S.E., Heselmeyer-Haddad, K., et al. 2013. Phylogenetic analysis of multiprobe fluorescence in situ hybridization data from tumor cell populations. *Bioinformatics* 29, i189–i198.

Chowdhury, S.A., Shackney, S.E., Heselmeyer-Haddad, K., et al. 2014. Algorithms to model single gene, single chromosome, and whole genome copy number changes jointly in tumor phylogenetics. *PLoS Comput. Biol.* 10, e1003740.

El-Kebir, M., Raphael, B.J., Shamir, R., et al. 2016. *Copy-Number Evolution Problems: Complexity and Algorithms*, pages 137–149. Springer International Publishing, Cham.

Fertin, G., Labarre, A., Rusu, I., et al. 2009. *Combinatorics of Genome Rearrangements*. MIT Press, Cambridge, MA.

Letouzé, E., Allory, Y., Bollet, M.A., et al. 2010. Analysis of the copy number profiles of several tumor samples from the same patient reveals the successive steps in tumorigenesis. *Genome Biol.* 11, R76.

McPherson, A., Wu, C., Wyatt, A.W., et al. 2012. nFuse: Discovery of complex genomic rearrangements in cancer using high-throughput sequencing. *Genome Res.* 22, 2250–2261.

Mohri, M. 2003. Edit-distance of weighted automata: General definitions and algorithms. *Int. J. Found. Comput. Sci.* 14, 957–982.

Mohri, M. 2004. Weighted finite-state transducer algorithms. An overview. *In Formal Languages and Applications*. pp. 551–563. Springer, Berlin-Heidelberg.

Oesper, L., Ritz, A., Aerni, S.J., et al. 2012. Reconstructing cancer genomes from paired-end sequencing data. *BMC Bioinformatics* 13 Suppl 6, S10.

Pinkel, D., Straume, T., and Gray, J.W. 1986. Cytogenetic analysis using quantitative, high-sensitivity, fluorescence hybridization. *Proc. Natl Acad. Sci. U. S. A.* 83, 2934–2938.

Savard, O.T., Gagnon, Y., Bertrand, D., et al. 2011. Genome halving and double distance with losses. *J. Comput. Biol.* 18, 1185–1199.

Schwarz, R.F., Trinh, A., Sipos, B., et al. 2014. Phylogenetic quantification of intra-tumour heterogeneity. *PLoS Comput. Biol.* 10, e1003535.

Shamir, R., Zehavi, M., and Zeira, R. 2016. A linear-time algorithm for the copy number transformation problem. *In* Grossi, R., and Lewenstein, M., eds, *27th Annual Symposium on Combinatorial Pattern Matching (CPM 2016)*, volume 54 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany.

Shao, M., and Lin, Y. 2012. Approximating the edit distance for genomes with duplicate genes under DCJ, insertion and deletion. *BMC Bioinformatics* 13, S13.

Tannier, E., Zheng, C., and Sankoff, D. 2009. Multichromosomal median and halving problems under different genomic distances. *BMC Bioinformatics* 10, 120.

The Cancer Genome Atlas Research Network. 2011. Integrated genomic analyses of ovarian carcinoma. *Nature* 474, 609–615.

Urban, A.E., Korbel, J.O., Selzer, R., et al. 2006. High-resolution mapping of DNA copy alterations in human chromosome 22 using high-density tiling oligonucleotide arrays. *Proc. Natl Acad. Sci. U. S. A.* 103, 4534–4539.

Address correspondence to:
Ron Zeira
Blavatnik School of Computer Science
Tel-Aviv University
Tel-Aviv 69978
Israel

E-mail: ronzeira@post.tau.ac.il

5. APPENDIX

5.1. Detailed case analysis

In this appendix, we present the details of the case analysis outlined in Section 3.4. We analyze the behavior of $M[i, d]$ as we increase d. We assume, by induction, that $f_j(d)$ is a piecewise linear function with parameters a_j, b_j and $base_j$ for $j = \text{prev}(i)$. Then, we examine several intervals that together contain all of the values that can be assigned to d. For each interval, we let d'_{opt} be an argument d' that minimizes $M[i, d]$ under the conditions of the examined case. Finally, we obtain the behavior of $f_i(d)$ in each interval, which is the behavior of the form we desire (i.e., $f_i(d)$ is a piecewise linear function defined by three segments). Denote $\max\{Q_i - \max\{d, d'\}, 0\}$ as $\arg_3$.

1. $Q_i \leq d_j^{min}$ (then, $\arg_3 = 0$):
   (a) $R_i \geq 0$:
      i. $d \leq a_j - R_i$:
         $d'_{opt} = a_j : f_i(d) = base_j$.
      ii. $a_j - R_i \leq d \leq b_j$:
         $d'_{opt} = d : f_i(d) = base_j + R_i - a_j + d$.
      iii. $b_j \leq d \leq d_j^{max}$:
         $d'_{opt} = d : f_i(d) = base_j + R_i - a_j - b_j + 2d$.
      iv. $d_j^{max} \leq d$:
         $d'_{opt} = d_j^{max} : f_i(d) = base_j + R_i - a_j - b_j + 2d$.

   (b) $R_i \leq 0$:
      i. $d \leq a_j + R_i$:
         $d'_{opt} = a_j : f_i(d) = base_j$.
      ii. $a_j + R_i \leq d \leq a_j + R_i$:
         $d'_{opt} = d : f_i(d) = base_j$.
      iii. $a_j \leq d \leq b_j$:
         $d'_{opt} = d : f_i(d) = base_j - a_j + d$.

iv. $b_j \le d \le b_j - R_i$:
$d'_{opt} = b_j : f_i(d) = base_j - a_j + d.$
v. $b_j - R_i \le d \le d_j^{max} - R_i$:
$d'_{opt} = b_j : f_i(d) = base_j + R_i - a_j - b_j + 2d.$
vi. $d_j^{max} - R_i \le d$:
$d'_{opt} = d_j^{max} : f_i(d) = base_j + R_i - a_j - b_j + 2d.$

2. $d_j^{min} \le Q_i \le a_j$:
(a) $R_i \ge 0$:
i. $Q_i \le a_j - R_i$: $\arg_3 = 0$ and the analysis is the same as in Case 1a.
ii. $a_j - R_i \le Q_i$:
A. $d \le a_j - R_i$:
$d'_{opt} = a_j : f_i(d) = base_j.$
B. $a_j - R_i \le d \le Q_i$:
$d'_{opt} = a_j : f_i(d) = base_j + R_i - a_j + d.$
C. $Q_i \le d$: $\arg_3 = 0$ and the rest of the analysis is the same as in Case 1a.

(b) $R_i \le 0$:
i. $Q_i \le a_j + R_i$: $\arg_3 = 0$ and the analysis is the same as in Case 1b.
ii. $a_j + R_i \le Q_i$:
A. $d \le a_j + R_i$:
$d'_{opt} = a_j : f_i(d) = base_j.$
B. $a_j + R_i \le d \le Q_i$:
$d'_{opt} = a_j : f_i(d) = base_j.$
C. $Q_i \le d$: $\arg_3 = 0$ and the rest of the analysis is the same as in Case 1b.

3. $a_j \le Q_i \le b_j$:
(a) $R_i \ge 0$:
i. $d \le a_j - R_i$:
$d'_{opt} = a_j : f_i(d) = base_j + Q_i - a_j.$
ii. $a_j - R_i \le d \le Q_i - R_i$:
$d'_{opt} = d + R_i : f_i(d) = base_j + Q_i - a_j.$
iii. $Q_i - R_i \le d \le Q_i$:
$d'_{opt} = d + R_i : f_i(d) = base_j + R_i - a_j + d.$
iv. $Q_i \le d \le b_j$:
$d'_{opt} = d : f_i(d) = base_j + R_i - a_j + d.$
v. $b_j \le d \le d_j^{max}$:
$d'_{opt} = d : f_i(d) = base_j + R_i - a_j - b_j + 2d.$
vi. $d_j^{max} \le d$:
$d'_{opt} = d_j^{max} : f_i(d) = base_j + R_i - a_j - b_j + 2d.$

(b) $R_i \le 0$:
i. $d \le a_j$:
$d'_{opt} = a_j : f_i(d) = base_j + Q_i - a_j.$
ii. $Q_i \le a_j - R_i$:
A. $a_j \le d \le Q_i$:
$d'_{opt} = d : f_i(d) = base_j + Q_i - a_j.$
B. $Q_i \le d \le a_j - R_i$:
$d'_{opt} = d : f_i(d) = base_j - a_j + d.$
iii. $a_j - R_i \le Q_i$:
A. $a_j \le d \le a_j - R_i$:
$d'_{opt} = d : f_i(d) = base_j + Q_i - a_j.$
B. $a_j - R_i \le d \le Q_i$:
$d'_{opt} = d : f_i(d) = base_j + Q_i - a_j.$
iv. $Q_i \le d \le b_j$:
$d'_{opt} = d : f_i(d) = base_j - a_j + d.$

v. $b_j \le d \le b_j - R_i$:
$d'_{opt} = b_j : f_i(d) = base_j - a_j + d.$

vi. $b_j - R_i \le d \le d_j^{max} - R_i$:
$d'_{opt} = b_j : f_i(d) = base_j + R_i - a_j - b_j + 2d.$

vii. $d_j^{max} - R_i \le d$:
$d'_{opt} = d_j^{max} : f_i(d) = base_j + R_i - a_j - b_j + 2d.$

4. $b_j \le Q_i \le d_j^{max}$:

(a) $R_i \ge 0$:

i. $d \le a_j - R_i$:
$d'_{opt} = a_j : f_i(d) = base_j + Q_i - a_j.$

ii. $Q_i - R_i \le a_j$:

A. $a_j - R_i \le d \le b_j - R_i$:
$d'_{opt} = d + R_i : f_i(d) = base_j + Q_i - a_j.$

B. $b_j - R_i \le d \le Q_i - R_i$:
$d'_{opt} = b_j : f_i(d) = base_j + Q_i + R_i - a_j - b_j + d.$

C. $Q_i - R_i \le d \le Q_i$:
$d'_{opt} = d : f_i(d) = base_j + Q_i + R_i - a_j - b_j + d.$

iii. $a_j \le Q_i - R_i \le b_j$:

A. $a_j - R_i \le d \le b_j - R_i$:
$d'_{opt} = d + R_i : f_i(d) = base_j + Q_i - a_j.$

B. $b_j - R_i \le d \le Q_i - R_i$:
$d'_{opt} = d + R_i : f_i(d) = base_j + Q_i + R_i - a_j - b_j + d.$

C. $Q_i - R_i \le d \le Q_i$:
$d'_{opt} = d : f_i(d) = base_j + Q_i + R_i - a_j - b_j + d.$

iv. $b_j \le Q_i - R_i$:

A. $a_j - R_i \le d \le b_j - R_i$:
$d'_{opt} = d + R_i : f_i(d) = base_j + Q_i - a_j.$

B. $b_j - R_i \le d \le Q_i - R_i$:
$d'_{opt} = d + R_i : f_i(d) = base_j + Q_i + R_i - a_j - b_j + d.$

C. $Q_i - R_i \le d \le Q_i$:
$d'_{opt} = d : f_i(d) = base_j + Q_i + R_i - a_j - b_j + d.$

v. $Q_i \le d$:
$d'_{opt} = Q_i : f_i(d) = base_j + R_i - a_j - b_j + 2d.$

(b) $R_i \le 0$:

i. $d \le a_j$:
$d'_{opt} = a_j : f_i(d) = base_j + Q_i - a_j.$

ii. $Q_i \le a_j - R_i$:

A. $a_j \le d \le Q_i$:
$d'_{opt} = a_j : f_i(d) = base_j + Q_i - a_j.$

B. $Q_i \le d \le a_j - R_i$:
$d'_{opt} = a_j : f_i(d) = base_j - a_j + d.$

C. $a_j - R_i \le d \le b_j - R_i$:
$d'_{opt} = a_j : f_i(d) = base_j - a_j + d.$

D. $b_j - R_i \le d \le Q_i - R_i$:
$d'_{opt} = b_j : f_i(d) = base_j + M_i - a_j - b_j + 2d.$

iii. $a_j - R_i \le Q_i \le b_j - R_i$:

A. $a_j \le d \le a_j - R_i$:
$d'_{opt} = a_j : f_i(d) = base_j + Q_i - a_j.$

B. $a_j - R_i \le d \le Q_i$:
$d'_{opt} = d + R_i : f_i(d) = base_j + Q_i - a_j.$

C. $Q_i \le d \le b_j - R_i$:
$d'_{opt} = d + R_i : f_i(d) = base_j - a_j + d.$

D. $b_j - R_i \le d \le Q_i - R_i$: $d'_{opt} = b_j : f_i(d) = base_j + M_i - a_j - b_j + 2d.$

iv. $b_j - R_i \le Q_i$:
A. $a_j \le d \le a_j - R_i$:
$d'_{opt} = a_j : f_i(d) = base_j + Q_i - a_j$
B. $a_j - R_i \le d \le b_j - R_i$:
$d'_{opt} = p + R_i : f_i(d) = base_j + Q_i - a_j.$
C. $b_j - R_i \le d \le Q_i$:
$d'_{opt} = b_j : f_i(d) = base_j + M_i + Q_i - a_j - b_j + d.$
D. $Q_i \le d \le Q_i - R_i$:
$d'_{opt} = b_j : f_i(d) = base_j + M_i - a_j - b_j + 2d.$
v. $Q_i - R_i \le d$:
$d'_{opt} = b_j : f_i(d) = base_j + M_i - a_j - b_j + 2d.$

5. $d_j^{max} \le Q_i$:
(a) $R_i \ge 0$:
i. $Q_i - R_i \le d_j^{max}$: The analysis is the same as in Case 4a for $d \le d_j^{max}$.
A. $d_j^{max} \le d \le Q_i$:
$d'_{opt} = d_j^{max} : f_i(d) = base_j + M_i + Q_i - a_j - b_j + d.$
ii. $Q_i - R_i \le d_j^{max}$:
A. $d \le b_j - R_i$:
$d'_{opt} = d + R_i : f_i(d) = base_j + Q_i - a_j.$
B. $b_j - R_i \le d \le d_j^{max} - R_i$:
$d'_{opt} = d + R_i : f_i(d) = base_j + M_i + Q_i - a_j - b_j + d.$
C. $d_j^{max} - R_i \le d \le d_j^{max}$:
$d'_{opt} = d_j^{max} : f_i(d) = base_j + M_i + Q_i - a_j - b_j + d.$
D. $d_j^{max} \le d \le Q_i - R_i$:
$d'_{opt} = d_j^{max} : f_i(d) = base_j + M_i + Q_i - a_j - b_j + d.$
E. $Q_i - R_i \le d \le Q_i$:
$d'_{opt} = d_j^{max} : f_i(d) = base_j + M_i + Q_i - a_j - b_j + d.$
iii. $Q_i \le d$:
$d'_{opt} = d_j^{max} : f_i(d) = base_j + M_i - a_j - b_j + 2d.$

(b) $R_i \le 0$:
i. The analysis of the cases obtained by adding the constraints defining Cases 4(b)ii, 4(b)iii, and 4(b)iv is similar.
ii. $d_j^{max} \le Q_i \le d_j^{max} - R_i$:
A. $d \le a_j - R_i$:
$d'_{opt} = a_j : f_i(d) = base_j + Q_i - a_j.$
B. $a_j - R_i \le d \le b_j - R_i$:
$d'_{opt} = b_j : f_i(d) = base_j + Q_i - a_j.$
C. $b_j - R_i \le d \le d_j^{max}$:
$d'_{opt} = d : f_i(d) = base_j + M_i + Q_i - a_j - b_j + d.$
D. $d_j^{max} \le d \le Q_i$:
$d'_{opt} = d + R_i : f_i(d) = base_j + M_i + Q_i - a_j - b_j + d.$
E. $Q_i \le d \le d_j^{max} - R_i$:
$d'_{opt} = d_j^{max} + R_i : f_i(d) = base_j + M_i - a_j - b_j + 2d.$
F. $d_j^{max} - R_i \le d$:
$d'_{opt} = d_j^{max} + R_i : f_i(d) = base_j + M_i - a_j - b_j + 2d.$
iii. $d_j^{max} - R_i \le Q_i$:
A. For $d \le d_j^{max}$, the analysis remains the same as in Case 5(b)ii.
B. $d_j^{max} \le d \le d_j^{max} - R_i$:
$d'_{opt} = d + R_i : f_i(d) = base_j + M_i + Q_i - a_j - b_j + d.$
C. $d_j^{max} - R_i \le d \le Q_i$:
$d'_{opt} = d_j^{max} : f_i(d) = base_j + M_i + Q_i - a_j - b_j + d.$
D. $Q_i \le d$:
$d'_{opt} = d_j^{max} + R_i : f_i(d) = base_j + M_i - a_j - b_j + 2d.$