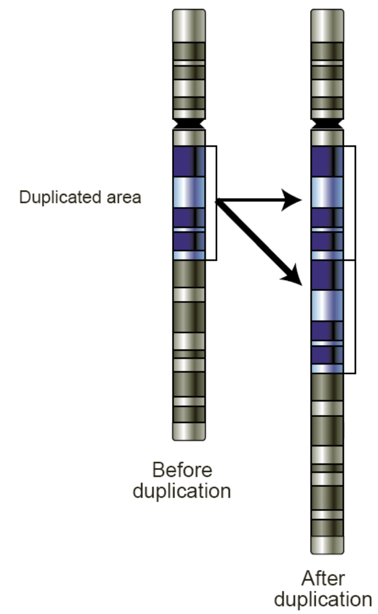
The Copy Number Transformation Problem



Meirav Zehavi
Tel Aviv University

Joint work with Ron Shamir and Ron Zeira

Copy Numbers (CNs)

CN Profile: A vector $V=(v_1,v_2,\ldots,v_n)$, where each v_i is a nonnegative integer.

- Example: (1,2,1,0,3,3).

Amplification: A triple $c = (l,h,1)$, where $1 \le l \le h \le n$.

- Effect (example): $c = (3,5,1)$;

$$c(1,2,1,0,3,3) = (1,2,2,0,4,3).$$

Deletion: A triple $c = (l,h,-1)$, where $1 \le l \le h \le n$.

- Effect (example): $c = (3,5,-1)$;

$$c(1,2,1,0,3,3) = (1,2,0,0,2,3).$$

Motivation

Genome rearrangement: The majority of the extant models either assume that each gene has a single copy or result in an NP-hard problem.

Detecting CN abnormalities: G-banding and fluorescence in situ hybridization (FISH); array comparative genomic hybridization (array CGH); next generation sequencing techniques.

Distances between CN profiles: Algorithmic aspects of questions related to these distances gained little scientific attention to date. We use the definition of Schwarz *et al.* (PLOS Comp. Biol., 2014).

Problem Statement

Let $S=(s_1,s_2,\ldots,s_n)$ and $T=(t_1,t_2,\ldots,t_n)$ be two CN profiles.

Copy Number Transformation (CNT) from *S* to *T*: A vector $C = (c_1,c_2,\ldots,c_m)$ of amplifications and deletions such that $c_m(c_{m-1}(\cdots(c_1(S))))=T$.

$C = (c_1,c_2,c_3)$

$$
\begin{array}{rcl}
S & = & (1, \boxed{1,}\, 1, 1, 1) \\
 & \downarrow & c_1 = (2,2,-1) \\
c_1(S) & = & (1, 0, 1, \boxed{1,}\, 1) \\
 & \downarrow & c_2 = (4,4,-1) \\
c_2(c_1(S)) & = & (\boxed{1, 0, 1, 0, 1}) \\
 & \downarrow & c_3 = (1,5,+1) \\
T = c_3(c_2(c_1(S))) & = & (2, 0, 2, 0, 2)
\end{array}
$$

Problem Statement

Let $S=(s_1,s_2,\ldots,s_n)$ and $T=(t_1,t_2,\ldots,t_n)$ be two CN profiles.

Copy Number Transformation (CNT) from *S* to *T*: A vector $C = (c_1,c_2,\ldots,c_m)$ of amplifications and deletions such that $c_m(c_{m-1}(\cdots(c_1(S))))=T$.

dist(*S*,*T*): The smallest size of a CNT from S to T.

The CNT Problem: Compute dist(S,T).

The Algorithm: Overview

The CNT problem can be solved in linear time and constant space.

Our approach:

- Prove four key propositions.
- An $O(nN^2)$-time, $O(N)$-space algorithm, DP-Alg, that is based dynamic programming.
- By using piecewise linear functions, we modify DP-Alg to obtain an $O(n)$-time, $O(1)$-space algorithm.

Key Propositions (Informal)

Proposition 1: It is sufficient to examine CNTs were all of the deletions precede all of the amplifications.

$$
\begin{aligned}
S &= (1, \boxed{1,}\, 1, 1, 1) \\
&\quad\downarrow \qquad\qquad c_1 = (2,2,-1) \\
c_1(S) &= (1, 0, 1, \boxed{1,}\, 1) \\
&\quad\downarrow \qquad\qquad c_2 = (4,4,-1) \\
c_2(c_1(S)) &= (\boxed{1, 0, 1, 0, 1}) \\
&\quad\downarrow \qquad\qquad c_3 = (1,5,+1) \\
T = c_3(c_2(c_1(S))) &= (2, 0, 2, 0, 2)
\end{aligned}
$$

Proposition 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. The same is true for amplifications.

Key Propositions (Informal)

If t_i>0 and d deletions affect s_i>0, then max{t_i-(s_i-d),0} amplifications should affect s_i.

Proposition 3: It is not necessary to store information indicating how many deletions/ amplifications affect s_i if t_i=0.

Proposition 4: The maximum number of deletions/amplifications that affect each s_i can be bounded by N.

Dynamic Programming

M[i,d] (1≤i≤n, 0≤d≤N): The size of an optimal transformation from $S^i=(s_1,s_2,\ldots,s_i)$ to $T^i=(t_1,t_2,\ldots,t_i)$ such that exactly d deletions affect s_i.

Recursive formula: $$\mathrm{M}[i,d] \leftarrow \min_{0 \le d' \le N} \{\mathrm{M}[\mathrm{prev}(i),d'] + \max\{d-d',0\} + \max\{a(i,d)-a(\mathrm{prev}(i),d'),0\} + \max\{Q_i - \max\{d,d'\},0\}\}.$$

$O(nN)$ entries + each entry is computed in time $O(N)$ → an $O(nN^2)$-time algorithm.

Piecewise Linear Functions

Main Idea: M can be described by $O(n)$ piecewise linear functions, where each function encapsulates $O(N)$ entries.

Each function is "well-behaved" – in particular, each function has only three linear segments.
→ The computation of an entry can be performed in time $O(1)$ rather than $O(N)$.

Each function can be represented in a compact manner.
→ the size of the table shrinks from $O(nN)$ to $O(n)$.

→ **Running time: $O(n)$;** space complexity: $O(1)$ (at each point of time, we store one piecewise linear function).


Thank You