

# Computational Complexity of Segmentation

Federico Adolfi (federico.adolfi@esi-frankfurt.de)

Ernst-Strüngmann Institute for Neuroscience in Cooperation with Max-Planck Society, Germany  
School of Psychological Science, University of Bristol, UK

Todd Wareham (harold@mun.ca)

Department of Computer Science, Memorial University of Newfoundland, Canada

Iris van Rooij (iris.vanrooij@donders.ru.nl)

Donders Institute for Brain, Cognition, and Behaviour, Radboud University, The Netherlands  
Department of Linguistics, Cognitive Science, and Semiotics & Interacting Minds Centre, Aarhus University, Denmark

## Abstract

Computational feasibility is a widespread concern that guides the framing and modeling of biological and artificial intelligence. The specification of cognitive system capacities is often shaped by unexamined intuitive assumptions about the search space and complexity of a subcomputation. However, a mistaken intuition might make such initial conceptualizations misleading for what empirical questions appear relevant later on. We undertake here computational-level modeling and complexity analyses of *segmentation* — a widely hypothesized subcomputation that plays a requisite role in explanations of capacities across domains — as a case study to show how crucial it is to formally assess these assumptions. We mathematically prove two sets of results regarding hardness and search space size that may run counter to intuition, and position their implications with respect to existing views on the subcapacity.

**Keywords:** segmentation; computational complexity; tractability; computational-level analysis; modeling; theory

## Introduction

Cognitive scientists routinely invoke subcapacities in decompositional efforts to reverse-engineer fully-fledged capacities of minds, brains and machines (Cummins, 2000; Miłkowski, 2013). For instance, speech processing is presumed to decompose into, among other things, segmentation and decoding, and action understanding into parsing, predicting and goal inference. These *subcomputations* are thought to tackle certain *problems* that the cognitive system faces to behave appropriately in the world.

Problems that originally show up in one domain (e.g., speech processing) are subsequently encountered in other domains (e.g., action understanding), and so the conceptual apparatus naturally carries over. As a result, cognitive scientists may come to view, e.g., the problem of segmenting speech as analogous to the problem of parsing actions. Researchers can then transfer ideas across the domains, adopting and adapting similar subcomputations in their explanations of the different capacities. What is passed along, however, will include latent (and possibly mistaken) notions about the computational properties of these problems as well. For instance, if a cognitive scientist believes that the search space of speech segmentation is large (combinatorially complex) and that this makes the problem hard, then by analogy, the same can be inferred about the parsing problem in action understanding.

Once such initial framing of a cognitive (sub)capacity is adopted, it completely shapes the kinds of empirical questions that appear relevant and in so doing determines the course of

research programs across disciplines and cognitive domains. Crucially, the assumptions that gave rise to the initial framing are seldom examined formally, and since they are taken for granted as background commitments, empirical tests are not designed to bear on them. These foundational oversights can sidetrack researchers into directions that will be largely immune to empirical corrective feedback later on.

To illustrate how crucial it is to formally assess the validity of intuitive assumptions about problem properties, and what can go astray if one doesn't, we undertake here a formal examination of an example subcapacity. Our case study is *Segmentation*<sup>1</sup>. This subcapacity figures ubiquitously in explanations of real-world cognitive capacities such as speech recognition, music perception, active sensing, event memory, temporal attention, action processing, and sequence learning. We focus on two classes of assumptions about its computational properties: i) the search space is excessively complex and ii) this makes the segmentation problem intrinsically hard.

To formally assess the theoretical viability of these assumed properties, we develop a formalization of the (intuitive) segmentation problem at Marr's (1982) computational level. Next, we submit this formalization to a mathematical analysis to assess the size of its search space, its computational hardness, and its possible sources of complexity using tools from computational complexity theory (Garey & Johnson, 1979; Arora & Barak, 2009; van Rooij, Blokpoel, Kwisthout, & Wareham, 2019). As our results may run counter to intuition, we end with a word of caution regarding the general non-intuitiveness of the computational properties of hypothesized cognitive problems.

## Conceptualization of segmentation

In order to rigorously examine computational assumptions, we need a mathematical formalization of the problem that can be submitted to further analyses. This computational-level model, in turn, should capture key aspects of the theorized cognitive capacity. To that end, in this section we synthesize conceptualizations of the segmentation problem as it appears in various cognitive domains.

<sup>1</sup>It relates closely to computations whose names vary depending on time period, cognitive domain, and theoretical framework: chunking, sampling, discretization, integration, grouping, packaging, quantization, sequencing, segregation, parsing, temporal pooling, temporal gestalt, boundary placement, temporal attention.

## Informal definitions: segmentation as a fundamental subcomputation

“How the brain processes sequences is a central question in cognitive science and neuroscience” (Jin, Lu, & Ding, 2020). A substantial amount of information available to the cognitive system is “continuous, dynamic and unsegmented” (Zacks et al., 2001). The purpose of the segmentation process is, then, “to generate elementary units of the appropriate temporal granularity for subsequent processing” (Giraud & Poeppel, 2012). Succinctly, “[t]he central nervous system appears to ‘chunk’ time” (Poeppel, 2003).

Several subfields of the cognitive and brain sciences have proposed segmentation as a key subcomputation. *Active listening* (cf. active sensing) casts it as “the selection of internal actions, corresponding to the placement of [...] boundaries” (Friston et al., 2021), “to sample the environment” (Poeppel & Assaneo, 2020). *Event cognition* similarly defines it as “the process of identifying event boundaries [...] a concomitant component of normal event perception” (Zacks et al., 2001). In *episodic memory*, it is “the process by which people parse the continuous stream of experience into events and sub-events [for] the formation of experience units” (Jeunehomme & D’Argembeau, 2018). Central to *music perception*, it features as determining the “perceptual boundaries of temporal gestalts” (Tenney & Polansky, 1980) and “entails the parsing into chunks” (Farbood, Rowland, Marcus, Ghitza, & Poeppel, 2015; Tillmann, 2012). The *speech recognition* literature describes it as the core process of “segmenting the continuous speech stream into units for further perceptual and linguistic analyses” (Teng, Cogan, & Poeppel, 2019), where it “allows the listener to transform [the] signal into segmented, discrete units, which form the input for subsequent decoding steps” (Poeppel & Assaneo, 2020). In *action processing*, “[a] fundamental problem observers must solve [...] is segmentation [...] Identifying distinct acts within the dynamic flow of motion is a basic requirement for engaging in further appropriate processing” (Baldwin, Andersson, Saffran, & Meyer, 2008).

Such ubiquitousness has been suggestive that the capacity “appeals to general principles the brain may use to solve a variety of problems” (Friston et al., 2021; Himberger, Chien, & Honey, 2018). “[M]any sequence-chunking tasks share common computational principles. [E.g.,] to find and encode the chunk boundaries” (Jin et al., 2020). Segmentation as a subcomputation appears across processing hierarchies as well, even when the world is relatively static: “[it] exists at multiple layers within a given problem” (Wyble & Bowman, 2019). The downstream operations on segments that partially determine optimal segmentation play similar roles but vary with cognitive domain and modeling framework.

*Segmentation*, concisely, is then a fundamental subcomputation whose requisite role across cognitive domains and processing hierarchies is to determine, given a sequence representation, the optimal boundary placement with respect to a downstream computation over segments.

## Formalization of segmentation

A succinct, yet informal, definition of segmentation can be stated by verbally specifying the inputs and outputs of the conjectured subcomputation.

SEGMENTATION (INFORMAL)

*Input:* A sequence and a downstream process that, for any given segment of the sequence, can evaluate its quality relative to domain-specific criteria.

*Output:* The best<sup>2</sup> segmentation of the sequence with respect to criteria relevant for the downstream process.

With this sketch in mind (see Fig. 1 for a schematic), we develop the formal definition of the computational-level model.

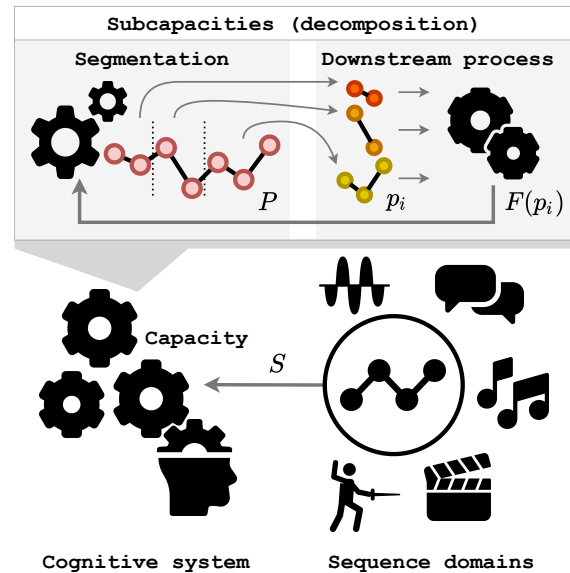


Figure 1: Segmentation is a core subcomputation in domains including sound, speech, music, action and event processing (bottom). Segmentation itself can be functionally decomposed such that domain-specific downstream processes inform which possible segmentations of a given sequence  $S$  are best (top). Refer to the main text for definitions of  $S$ ,  $P$ ,  $F$  and  $p_i$ .

We envision an input sequence  $S = (s_1, s_2, \dots, s_n)$  that captures the idea of a time-ordered representation the cognitive system must work with. Its origin could be sensory encoding at the periphery or deeper, more elaborate processes alike (e.g., an encoding of the acoustic envelope of speech or music, or a compressed representation of a visual scene). As instances of the segmentation problem appear throughout processing hierarchies, their inputs vary in origin and nature. We model the sequence with according generality. Next, we pin down the notion of a downstream cognitive process that computes over segments  $p \in \mathcal{P}$  (e.g., a decoder that maps speech segments to phonemes, or a module that maps scene segments to action meanings). Our formalization is agnostic as to what

<sup>2</sup>Without loss of generality, here ‘best’ could be replaced by ‘good enough’, and our formal results would still apply.

these domain-specific processes, and theoretical frameworks used to model them, might be. We aim for generality and simply model, with a function  $F : \mathcal{P} \mapsto \mathbb{Z}^+$  available at the input, the idea that the process is capable of guiding the placement of boundaries. This is achieved by reporting back some (discretized) aspect of its performance  $F(p) \in \mathbb{Z}^+$  (e.g., label probability, likelihood w.r.t. generative model, depending on framework). The desired output — a useful segmentation scheme — is modeled as a collection  $P$  of disjoint segments jointly making up the sequence, whose overall appropriateness  $V(P)$  with respect to the downstream process is optimal. These modeling choices yield the following formalization.

#### SEGMENTATION (FORMAL)

*Input:* a finite sequence  $S = (s_1, s_2, \dots, s_N)$  of length  $N \in \mathbb{N}$ , with  $s_i \in \mathbb{Z}$  and a scoring function  $F : \mathcal{P} \mapsto \mathbb{Z}^+$  that maps contiguous subsequences  $p = (s_i, s_{i+1}, \dots, s_{i+q})$  to a positive value  $F(p)$ .

*Output:* a segmentation of  $S$  into contiguous subsequences,  $P = \{(s_1, s_2, \dots), \dots, (\dots, s_{n-1}, s_N)\}$ , where  $\forall p_i, p_j \in P : I(p_i) \cap I(p_j) = \emptyset$  and  $\|_1^{|P|} p_i = S$ , such that its overall value  $V(P) = \sum_{p \in P} F(p)$  is maximum.

(We write  $I(p)$  to denote the set of element indices of subsequence  $p$  in the original sequence and  $\|_a^b p_i$  to denote concatenation of subsequences  $p_a$  through  $p_b$ .)

### Assumptions about segmentation

To determine the course of our analyses, we survey views on the computational properties of the segmentation problem. We illustrate with examples and synthesize core intuitions.

#### Problem properties: segmentation as a computational challenge

**Hardness and complexity.** Segmentation problems have been widely assumed to be computationally challenging. This is evidenced in explicit statements and in the ‘solutions’ researchers propose after taking onboard certain beliefs about hardness. To illustrate: “Speech recognition is not a simple problem. The auditory system must parse a continuous signal into discrete words” (Friston et al., 2021). “It is hard for a brain, and very hard for a computer” (Poeppel, 2003). “[S]egmentation requires inference over the intractably large discrete combinatorial space of partitions.” (Franklin, Norman, Ranganath, Zacks, & Gershman, 2020).

**Sources of complexity.** As is evident in researchers’ descriptions, the hardness is attributed to the (presumed) combinatorial explosion involved in the number of possible segmentation schemes — the size of the problem search space is informally taken as the source of computational complexity. Again, to illustrate: “Where should these candidate boundaries be placed? In an extreme case, we could place boundaries at every combination of time points [...] but that would be computationally inefficient given that we can reduce the scope of possibilities” (Friston et al., 2021). “The prob-

lem would be enormously complicated by the presence of so many candidates [...]” (Brent, 1999).

**Solutions for complexity.** Arguably as a consequence of coupling these intuitions with additional assumptions, the effectiveness of certain solutions has been taken for granted. “From the computational perspective, the aim of research in segmentation [...] is to identify mechanisms [that] reduce these computational burdens by reducing the number of candidate[s]” (Brent, 1999). This position has motivated the search for bottom-up segmentation cues or top-down biases (e.g., priors) that would achieve, among other things, such a narrowing down (e.g., Teng et al., 2019; Friston et al., 2021). “We suggest a different role [of cues] in which they are part of the [segmentation] (rather than decoding) process” (Ghitza, 2012). For instance, researchers may observe environmental (Ding et al., 2017) and neural (Teng, Tian, Rowland, & Poeppel, 2017; Teng, Tian, Doelling, & Poeppel, 2017) regularities suggestive of segment-size constrained segmentation processes (Poeppel & Assaneo, 2020; Poeppel, 2003).

#### Core assumptions

This survey reveals a core set of intuition-based assumptions about the computational properties of segmentation:

- Real-world sequences (e.g., speech, music, scenes, actions) and internal representations alike (e.g., memories of experiences) are “complex, continuous, dynamic flows”.
- The cognitive system needs to make use of discrete representations of segments that are appropriate (size- and content-wise) for downstream tasks.
- The problem is “hard” — the obstacle being that there are “too many” possible segmentations of a given sequence.
- Cognitive systems must reduce the possibilities somehow, e.g., via bottom-up cues and/or top-down biases.

#### Computational complexity of segmentation

It is generally non-obvious what problems are genuinely (as opposed to merely apparently) hard, which refinements will render a model tractable, or which restrictions will effectively reduce a search space. Intuitions about computational properties of problems are frequently mistaken, hence need to be validated against formal analyses (van Rooij, Evans, Müller, Gedge, & Wareham, 2008). This section presents a complexity analysis in two parts according to the assumed properties they examine: search space size, and problem hardness.

#### Search space of segmentation

We analyze the search space size as a possible source of hardness by envisioning a simple brute-force algorithm. If the number of candidate solutions grows polynomially (i.e., upperbounded by  $N^c$ , where  $N$  is the sequence length and  $c$  is some constant), then such an algorithm would be tractable. We describe this growth through combinatorial analysis; first for the unconstrained problem and then including various theoretically motivated constraints.

**Unbounded parts.** When the size  $q$  of the segments is not constrained other than by the length  $N$  of the sequence, i.e.,  $q \in [1, N]$ , all boundary placements are possible. Notice there is a bijection between binary strings of length  $N - 1$  and boundary placements in sequences of length  $N$  (Fig. 2).

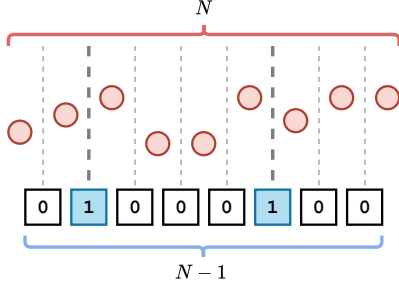


Figure 2: Binary strings encode boundary placements. A sequence of length  $N$  (top) admits  $N - 1$  choices of the presence/absence of boundaries (bottom).

Since the number of possible binary strings of length  $k$  is given by  $2^k$ , the number of possible segmentations that use unbounded parts grows as  $O(2^{N-1})$  (i.e., exponentially).

**Segmentation as integer composition.** In order to incorporate various constraints in combinatorial analyses, we draw an analogy between segmentation and *integer compositions* — a puzzle in combinatorics. This enables us to take an analytic combinatorics approach (Flajolet & Sedgewick, 2009) to the latter and leverage the results to infer properties of the former.

**Definition 1** (Integer composition). A composition of an integer  $N$  is an ordered list  $C = (p_1, p_2, \dots, p_k)$  of positive integer parts  $p_i \in \mathbb{N}^+$ , such that  $N = \sum_{p \in C} p$ .

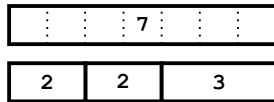


Figure 3: Segmentation as integer composition. An integer (top), arbitrary parts, and their composition (bottom) stand for a sequence, segments, and a possible segmentation.

To obtain the growth rate for various restricted cases, we derive generating functions for each, whose coefficients count the number of compositions, and submit them to analysis based on the following lemma.

**Lemma 1** (Growth rate of the coefficients of a rational function). Let  $S(x) = \sum_{n \geq 0} s_n x^n = \frac{P(x)}{Q(x)}$  be a rational function with  $Q(0) \neq 0$  and assume  $P(x)$  and  $Q(x)$  do not have any roots in common. The general form of the coefficients is  $[x^N]S(x) = A^N \Phi(N)$ , where  $A^N$  is the exponential growth factor and  $\Phi(N)$  is the subexponential growth factor. Then the exponential growth rate  $A$  of the sequence of coefficients  $(s_n)$  is equal to  $|\frac{1}{\alpha}|$ , where  $\alpha$  is the root of  $Q(x)$  of smallest modulus (for proof, see Bóna, 2016, Theorem 7.10).

**Lower-bounded parts.** We consider integer compositions involving parts  $p_i \in [a, N]$ , with  $1 < a < N$ .

**Theorem 1.** The number of  $[a, N]$ -restricted integer compositions of  $N$  grows exponentially with  $N$ .

*Proof.* For each part  $p_i$  the choice is among the positive integers  $(a, a + 1, a + 2, \dots)$ , so in terms of the generating function we have  $(x^a + x^{a+1} + \dots)$ , where  $a$  is an integer constant. By factoring and using the closed form of the geometric series we write  $x^a(1 + x + x^2 + \dots) = x^a \frac{1}{1-x}$ . For a  $k$ -part composition we thus have  $(\frac{x^a}{1-x})^k$ , so for compositions of any number of parts we sum<sup>3</sup> over  $k$  and write  $\sum_{k=1}^{\infty} (\frac{x^a}{1-x})^k$ , which similarly as above can be simplified  $\frac{x^a}{1-x} [1 + (\frac{x^a}{1-x})^1 + (\frac{x^a}{1-x})^2 + \dots]$  until we arrive at the closed form that completes the construction:

$$G_{LB}(x) := \frac{x^a}{1-x-x^a}$$

Having derived the generating function we are interested in the growth rate of the coefficients  $[x^n]G_{LB}(x)$ . Let  $Q(x) = 1 - x - x^a$ , and note that  $Q(0) = 1$  and  $\lim_{x \rightarrow 1} Q(x) = -1$ . Since  $Q(0) > 0$  and  $Q(1 - \epsilon) < 0$  for some small positive  $\epsilon$ , by the intermediate value theorem, it follows that there exists a root that satisfies  $0 < \alpha < 1$ . By Lemma 1 the exponential growth factor is  $A^N$ , for a constant  $A > 1$ . ■

**Upper-bounded parts.** We consider integer compositions involving parts  $p_i \in [1, b]$ , with  $1 < b < N$ .

**Theorem 2.** The number of  $[1, b]$ -restricted integer compositions of  $N$  grows exponentially with  $N$ .

*Proof.* For each part  $p_i$  the choice is among the positive integers  $(1, 2, 3, \dots, b)$ , so in terms of the generating function we have  $(x^1 + x^2 + \dots + x^b)$ , where  $b$  is an integer constant. Factoring out  $x$  and using the identity  $1 + y + y^2 + \dots + y^n = \frac{1-y^{n+1}}{1-y}$ , we write  $x(1 + x^1 + \dots + x^{b-1}) = \frac{x-x^b}{1-x}$ . For  $k$ -part compositions we raise to the  $k$ -th power and for compositions of any number of parts we sum over all  $k$ , which yields  $\sum_{k=0}^{\infty} (\frac{x-x^b}{1-x})^k = 1 + (\frac{x-x^b}{1-x})^1 + (\frac{x-x^b}{1-x})^2 + \dots$ , and finally by using the geometric series and simplifying we complete the construction:

$$G_{UB}(x) := \frac{1-x}{1-2x+x^{b+1}}$$

Having derived the generating function we are interested in the growth rate of the coefficients  $[x^n]G_{UB}(x)$ . Following the intermediate value theorem, let  $Q(x) = 1 - 2x + x^{b+1}$  and note that  $Q(0) > 0$  and  $Q(\frac{3}{4}) < 0$ , for any  $b$  as above, hence there exists a root  $\alpha$  located in the interval  $(0, \frac{3}{4}]$ . By Lemma 1 the exponential growth factor is  $A^N$ , for a constant  $A > 1$ . ■

**Doubly-bounded parts.** We consider compositions involving parts  $p_i \in [a, b]$ , with  $1 < a < b < N$ .

<sup>3</sup>Note that summing from  $k = 0$  or from  $k = 1$  to  $n$  does not affect the denominator of the resulting generating function, therefore the results will be robust to these variations.

**Theorem 3.** The number of  $[a, b]$ -restricted integer compositions of  $N$  grows exponentially with  $N$ .

*Proof.* For each part  $p_i$  the choice is among the positive integers  $(a, a+1, \dots, b)$ , so in terms of the generating function we have  $(x^a + x^{a+1} + \dots + x^b)$ . Factoring out  $x^a$  and using  $1 + y + y^2 + \dots + y^n = \frac{1-y^{n+1}}{1-y}$ , we write  $\frac{x^a - x^{b+1}}{1-x}$ . For compositions of any number  $k$  of parts, we have  $\sum_{k=1}^{\infty} (\frac{x^a - x^{b+1}}{1-x})^k = 1 + (\frac{x^a - x^{b+1}}{1-x})^1 + (\frac{x^a - x^{b+1}}{1-x})^2 + \dots$  Using the geometric series and simplifying, we arrive at the final form:

$$G_{DB}(x) := \frac{x^a - x^{b+1}}{1 - x - x^a + x^{b+1}}$$

Having derived the generating function we are interested in the growth rate of the coefficients  $[x^n]G_{DB}(x)$ . Since  $1 < a < b$ , with  $c = b - a \geq 2$ , we write the polynomial in the denominator  $Q(x) = 1 - x - x^a + x^{a+c}$  which we then rewrite as  $(1-x) + x^a(x^c - 1) = (1-x)(1 - x^a[x^{c-1} + x^{c-2} + \dots + 1])$ . Note that  $Q(x)$  has a root in  $(0, 1)$  if and only if  $q(x) = 1 - x^a(x^{c-1} + x^{c-2} + \dots + 1)$  has such a root. We have that  $q(0) = 1$  and furthermore  $\lim_{x \rightarrow 1} q(x) = 1 - c$ , and recall  $c \geq 2$ . By the intermediate value theorem, since  $q(0) > 0$  and  $q(1 - \epsilon) < 0$  for some small positive  $\epsilon$ , there exists a root  $\alpha$  of  $Q(x)$  in the open interval  $(0, 1)$  for any integers  $a, b$  constrained as above. By Lemma 1 the exponential growth factor is  $A^N$ , for a constant  $A > 1$ . ■

## Hardness of segmentation

We showed that intuitive constraints do not render brute-force segmentation tractable. One may be tempted to conclude that this demonstrates the conjectured hardness of the segmentation problem. However, in this section we present a theorem that contradicts this conclusion. The proof builds on the technique of (polynomial-time) reduction (Arora & Barak, 2009; Garey & Johnson, 1979; van Rooij et al., 2019).

**Definition 2.** (Reduction) Let  $A$  and  $B$  be computational problems. We say  $A$  is *reducible* to  $B$  if a tractable algorithm for  $B$  (in case it exists) could also be used to solve  $A$  tractably. An algorithm for transforming instances of one problem into those of another is called a *reduction*.

We present a reduction from the problem SEGMENTATION to a problem in graph theory. On the way, we will have introduced an alternative way of thinking about segmentation at the computational and algorithmic levels.

**Theorem 4.** SEGMENTATION is tractable (polynomial-time computable) in the absence of constraints.

*Proof.* We will show that, given an arbitrary instance of the segmentation problem, we can tractably construct an instance (with the correct associated output) of a target problem which is itself tractably computable. To begin, we introduce a class of graphs which we use as a stepping stone.

**Definition 3** (Interval Graph). An interval graph is an undirected graph  $G = (V, E)$  built from a collection of

intervals  $\{p_i\}_{i=1}^k$  by creating one vertex  $v_i$  for each interval  $p_i$  and an edge  $(v_i, v_j)$  whenever the corresponding intervals have a non-empty intersection:  $E = \{(v_i, v_j) \in V \times V \mid p_i \cap p_j \neq \emptyset\}$ .

Consider an instance of SEGMENTATION. Given an input sequence, it is possible to construct an *interval graph* that satisfies Def. 3. Algorithm 1 demonstrates the procedure.

**Algorithm 1** Construct segment graph from sequence.

---

*Input:*  
 $S = (s_1, s_2, s_3, \dots, s_N)$ ,  $s_i \in \mathbb{Z}$  ▷ sequence  
 $F : \mathcal{P} \mapsto \mathbb{Z}^+$  ▷ scoring function

*Output:*  
 $V = \{(v_1, w_1), \dots, (v_q, w_q)\}$  ▷ weighted vertex set  
 $E = \{(v_i, v_j), \dots\}$  ▷ edge set

---

```

1: function BUILDSEGMENTGRAPH( $S, F$ )
2:    $P \leftarrow []$  ▷ legal segments
3:    $V \leftarrow \{\}$  ▷ weighted vertex set
4:    $E \leftarrow \{\}$  ▷ edge set
▷ construct weighted vertex set
5:   for  $i \leftarrow 1$  to  $|S|$  do
6:     for  $j \leftarrow 0$  to  $|S| - i$  do
7:        $segment \leftarrow S[i : i + j]$ 
8:        $weight \leftarrow F(segment) \times (-1)$ 
9:        $P.append(segment)$ 
10:       $V.append((|P|, weight))$ 
11:    end for
12:  end for
▷ construct edge set
13:  for  $i \leftarrow 1$  to  $|P| - 1$  do
14:    for  $j \leftarrow i + 1$  to  $|P|$  do
15:      if  $P[i] \cap P[j] \neq \emptyset$  then
16:         $E.append((i, j))$ 
17:      end if
18:    end for
19:  end for
20:  return  $(V, E)$ 
21: end function

```

---

*Remark.* Algorithm 1 involves systematically generating all legal segments, computing and negating their weights, checking their pairwise overlap, and using this to construct a graph. We call this object a *segment graph*.

Consider the *time complexity* of Algorithm 1. The elementary instructions are the weight computation (line 8), appending (lines 9-10, 16), and set intersection (line 15); all of which are polynomial-time computable ( $F$  is assumed to be). We focus now on the number of implied iterations. The loops defined in lines 5-6 yield  $N + (N - 1) + \dots + 1$  iterations (the number of possible segments), given by a polynomial:

$$|P_N| = \sum_{k=1}^N k = \frac{N(N+1)}{2} \quad (N^{\text{th}} \text{ triangular number})$$

The loops defined in lines 13-14 yield a number of iterations equal to the number of segment pairs  $(p_i, p_j) \in P_N^*$ , given by the binomial coefficient  $\binom{n}{k}$  with  $n = |P_N|$  and  $k = 2$ , which grows as a quadratic in  $|P_N|$  (i.e. 4<sup>th</sup>-degree polynomial in  $N$ ).

$$|P_N^*| = \binom{|P_N|}{2} \sim O(N^4)$$

This algorithmic analysis demonstrates that BUILDSEGMENTGRAPH (Algorithm 1) is polynomial-time computable.

Consider next the *correctness* of Algorithm 1. We will show that a segment graph encodes the properties of candidate solutions to an instance of SEGMENTATION. For this, we need the following definitions.

**Definition 4** (Independent sets and maximality). Let  $G = (V, E)$  denote a graph. We call a vertex set  $V^* \subseteq V$  an *independent set* if there exist no two vertices  $u, v \in V^*$  such that  $(u, v) \in E$ . Such a set is said to be *maximal* if there exists no vertex  $v \in V$  that can be added to  $V^*$  without breaking the independence.

**Definition 5** (Dominating sets and minimality). Let  $G = (V, E)$  denote a graph. We call a vertex set  $V^* \subseteq V$  a *dominating set* if for all  $v \in V$ , either  $v \in V^*$  or there is an edge  $(v, u) \in E$  for some  $u \in V^*$ . Such a set is said to be *minimal* if there exists no vertex  $v \in V^*$  that can be removed without breaking the dominance.

By construction, a legal segmentation (i.e., a collection of disjoint segments whose concatenation yields the original sequence) is guaranteed to be represented within the segment graph as a subset of vertices with two properties:

- *maximal independence*: vertices are pairwise non-adjacent because segments in a segmentation should be disjoint; since the segments should span the sequence, adding any vertex breaks independence.
- *minimal dominance*: vertices in the graph are either in the subset or adjacent to one of its elements because once a segment subset spans the sequence, any other segment is guaranteed to overlap; since the segments should be disjoint, removing any vertex breaks dominance.

*Remark.* How segment graphs make the structure of the original sequence problem transparent is illustrated in Fig. 4.

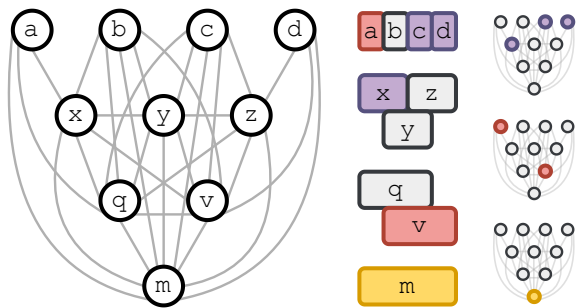


Figure 4: Segment graph encoding (left) of a sequence of length 4. Nodes correspond to possible segments and edges represent pairwise overlap in the sequence. Possible segments are grouped according to length and overlap (middle). Candidate solutions are vertex subsets in the segment graph (right). Three example segmentations are color coded.

A general feature of dominance and independence on arbitrary graphs is useful:

**Lemma 2.** An independent vertex set in a graph is a dominating set if and only if it is a *maximal* independent set. Any such set is necessarily also a *minimal* dominating set. (cf. Berge, 1962; Goddard & Henning, 2013).

It follows from the above and Lemma 2 that if a vertex subset in a segment graph is independent and dominant, then it is a candidate solution (i.e., valid segmentation). A feasible solution has, additionally, minimum weight among candidates: it is a *minimum-weight independent dominating set*. With this, we introduce the formal graph problem we reduce to.

**Definition 6.**

MINIMUM-WEIGHT INDEPENDENT DOMINATING SET  
*Input:* A vertex-weighted graph  $G = (V, E)$ . For each  $v \in V$  we have a weight  $W(v) \in \mathbb{Z}$ .  
*Output:* An independent dominating set  $V^* \subseteq V$  such that  $Q(V^*) = \sum_{v \in V^*} W(v)$  is minimum.

So far, we have established that, given an instance  $I_{seq}$  of SEGMENTATION, we can construct, in polynomial time by Algorithm 1, call it  $A(\cdot)$ , a corresponding instance  $I_{graph} = A(I_{seq})$  of MINIMUM-WEIGHT INDEPENDENT DOMINATING SET. This demonstrates the validity of the reduction and we now finally consider the tractability of the problems.

Though the problem of finding minimum-weight independent dominating sets is *NP-hard* in general and remains so in several special cases (Garey & Johnson, 1979; Liu, Poon, & Lin, 2015), the following input restriction is relevant.

**Lemma 3.** MINIMUM-WEIGHT INDEPENDENT DOMINATING SET is polynomial-time computable provided the input graph is an *interval graph*. (for proof, see Chang, 1998, Theorem 2.4).

Recall that the restriction required by Lemma 3 is guaranteed by our reduction. Hence, we conclude SEGMENTATION is tractably computable, which completes the proof. ■

## Discussion

Computational feasibility is a widespread concern that motivates choices in the framing and modeling of biological and artificial intelligence. While implicit or informal assumptions abound, the reality may turn out to be counterintuitive as they are examined formally. Here, we undertook a formal examination of the existing computational assumptions about *Segmentation*. Using complexity-theoretic tools, we mathematically proved two sets of results that run counter to commonly-held assumptions: 1) the search space is either not large to begin with or it is large but placing intuitive constraints does not alleviate the issue; and 2) a computational model of segmentation that formalizes its conceptualization across domains is tractably computable in the absence of widely-adopted constraints to address the assumed hardness.

Beyond our proofs, we set the groundwork for further refinements of segmentation theory and its computational analyses: a) we contributed a formalization of the computa-

tion that satisfies a domain-agnostic specification; b) we illustrated the relationship between segmentation and integer compositions, which makes the search space amenable to asymptotic analyses; and c) we built a bridge from the problem as originally defined on sequences to the mathematics of graphs, which opens up alternative formalisms to model it and to think about it algorithmically. A desirable consequence of translating problems between formal domains is that structure which was originally hidden from view may become visible.

Our results challenge existing intuitions about hardness of the segmentation problem and its sources of complexity, and by extension question the motivation of proposed solutions and their associated empirical research foci. For instance, concerns about search space size and what mitigates it may be misplaced. The space of possible segments is not exponential to begin with; the space of segmentations is. However, the bounds on segment size are, neither individually nor combined, a source of exponentiality. Left unexamined, this may still appear to support the conjectured hardness of the problem. But our tractability proof challenges this intuitive conclusion. It demonstrates that no assumptions about bottom-up segmentation cues or top-down biases on segment properties are necessary to make the formal problem tractable. These proofs run counter to the computational efficiency concerns that partially motivate segmentation theories. For instance, proposals that argue from minimal units of representation (cf. Pöppel, 1997), temporal integration limits of neuronal populations (cf. Overath, McDermott, Zarate, & Poeppel, 2015), intrinsic oscillatory timescales (cf. Ghitza, 2012; Wolff et al., 2022), bottom-up segmentation cues (e.g., Giraud & Poeppel, 2012), and top-down biases on candidate search (e.g., Friston et al., 2021), which to some extent build on the supposition of problem hardness, search space size, and various sources of complexity. This suggests that intractability concerns, if any, might be better placed, for instance, on the processes guiding segmentation rather than the boundary placement itself.

Together, the results proven here caution against intuitive notions about the properties of computational problems driving empirical programs, and demonstrate the need and benefits of critically assessing their soundness. Whenever intuitions are challenged, this enables researchers to either slightly or entirely redirect efforts as ideas shift regarding what evidence is relevant to collect. For instance, if researchers believe that a certain problem is computationally hard and that some set of neural and environmental regularities might speak to constraints that make it tractable, then they would be inclined to look for those regularities that satisfy such a requirement. If, however, the original belief is removed, the target regularities or the kinds of experiments that are adequate to test their putative role might be different.

We close with a similar word of caution about interpreting our results. These are to some degree tied to the particular formalization we put forth. While modeling choices were motivated and they bear some generality, alternative theoretical commitments are conceivable. For instance, an

extended model could allow for multiple unsegregated high-dimensional input streams; it is an open question whether it would have different complexity properties. We view our analyses not as the last word on the computational complexity of segmentation but rather as initial words in a conversation with a sound formal basis.

## Acknowledgments

We thank the Computational Cognitive Science group at the Donders Institute for Brain, Cognition, and Behaviour for discussions, Nils Donselaar for invaluable feedback on a previous version that helped improve the manuscript, and Ronald de Haan for comments on future directions of this work. FA thanks David Poeppel for support and discussions on auditory segmentation. TW was supported by NSERC Discovery Grant 228104-2015.

## References

- Arora, S., & Barak, B. (2009). *Computational complexity: A modern approach*. Cambridge ; New York: Cambridge University Press.
- Baldwin, D., Andersson, A., Saffran, J., & Meyer, M. (2008). Segmenting dynamic human action via statistical structure. *Cognition*, 106(3), 1382–1407.
- Berge, C. (1962). *The theory of graphs and its applications*. Methuen.
- Bóna, M. (2016). *Introduction to enumerative and analytic combinatorics* (Second edition ed.). Boca Raton: CRC Press, Taylor & Francis Group.
- Brent, M. R. (1999). Speech segmentation and word discovery: A computational perspective. *Trends in Cognitive Sciences*, 3(8), 294–301.
- Chang, M.-S. (1998). Efficient Algorithms for the Domination Problems on Interval and Circular-Arc Graphs. *SIAM J. Comput.*, 27(6), 1671–1694.
- Cummins, R. (2000). How does it work? versus what are the laws?: Two conceptions of psychological explanation. In *Explanation and cognition* (pp. 117–144).
- Ding, N., Patel, A. D., Chen, L., Butler, H., Luo, C., & Poeppel, D. (2017). Temporal modulations in speech and music. *Neuroscience & Biobehavioral Reviews*, 81, 181–187.
- Farbood, M. M., Rowland, J., Marcus, G., Ghitza, O., & Poeppel, D. (2015). Decoding time for the identification of musical key. *Attention, Perception, & Psychophysics*, 77(1), 28–35.
- Flajolet, P., & Sedgewick, R. (2009). *Analytic combinatorics*. Cambridge ; New York: Cambridge University Press.
- Franklin, N. T., Norman, K. A., Ranganath, C., Zacks, J. M., & Gershman, S. J. (2020). Structured Event Memory: A neuro-symbolic model of event cognition. *Psychological Review*, 127(3), 327–361.
- Friston, K. J., Sajid, N., Quiroga-Martinez, D. R., Parr, T., Price, C. J., & Holmes, E. (2021). Active listening. *Hearing Research*, 399, 107998.

- Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. New York: W. H. Freeman.
- Ghitza, O. (2012). On the Role of Theta-Driven Syllabic Parsing in Decoding Speech: Intelligibility of Speech with a Manipulated Modulation Spectrum. *Front. Psychol.*, 3.
- Giraud, A.-L., & Poeppel, D. (2012). Cortical oscillations and speech processing: Emerging computational principles and operations. *Nature Neuroscience*, 15(4), 511–517.
- Goddard, W., & Henning, M. A. (2013). Independent domination in graphs: A survey and recent results. *Discrete Mathematics*, 313(7), 839–854.
- Himberger, K. D., Chien, H.-Y., & Honey, C. J. (2018). Principles of Temporal Processing Across the Cortical Hierarchy. *Neuroscience*.
- Jeunehomme, O., & D'Argembeau, A. (2018). Event segmentation and the temporal compression of experience in episodic memory. *Psychological Research*.
- Jin, P., Lu, Y., & Ding, N. (2020). Low-frequency neural activity reflects rule-based chunking during speech listening. *eLife*, 9, e55613.
- Liu, C.-H., Poon, S.-H., & Lin, J.-Y. (2015). Independent dominating set problem revisited. *Theoretical Computer Science*, 562, 1–22.
- Marr, D. (1982). *Vision: A computational investigation into the human representation and processing of visual information*. Cambridge, Mass: MIT Press.
- Miłkowski, M. (2013). Reverse-engineering in Cognitive Science. In K. Talmont-Kaminski & M. Miłkowski (Eds.), *Regarding the Mind, Naturally: Naturalist Approaches to the Sciences of the Mental*. Cambridge Scholars Publishing.
- Overath, T., McDermott, J. H., Zarate, J. M., & Poeppel, D. (2015). The cortical analysis of speech-specific temporal structure revealed by responses to sound quilts. *Nature Neuroscience*, 18(6), 903–911.
- Poeppel, D. (2003). The analysis of speech in different temporal integration windows: Cerebral lateralization as 'asymmetric sampling in time'. *Speech Communication*, 41(1), 245–255.
- Poeppel, D., & Assaneo, M. F. (2020). Speech rhythms and their neural foundations. *Nat Rev Neurosci*, 21(6), 322–334.
- Pöppel, E. (1997). A hierarchical model of temporal perception. *Trends in Cognitive Sciences*, 1(2), 56–61.
- Teng, X., Cogan, G. B., & Poeppel, D. (2019). Speech fine structure contains critical temporal cues to support speech segmentation. *NeuroImage*, 202, 116152.
- Teng, X., Tian, X., Doelling, K., & Poeppel, D. (2017). Theta band oscillations reflect more than entrainment: Behavioral and neural evidence demonstrates an active chunking process. *European Journal of Neuroscience*.
- Teng, X., Tian, X., Rowland, J., & Poeppel, D. (2017). Concurrent temporal channels for auditory processing: Oscillatory neural entrainment reveals segregation of function at different scales. *PLOS Biology*, 15(11), e2000812.
- Tenney, J., & Polansky, L. (1980). Temporal Gestalt Perception in Music. *Journal of Music Theory*, 24(2), 205.
- Tillmann, B. (2012). Music and Language Perception: Expectations, Structural Integration, and Cognitive Sequencing. *Topics in Cognitive Science*, 4(4), 568–584.
- van Rooij, I., Blokpoel, M., Kwisthout, J., & Wareham, T. (2019). *Cognition and intractability: A guide to classical and parameterized complexity analysis*. Cambridge University Press.
- van Rooij, I., Evans, P., Müller, M., Gedge, J., & Wareham, T. (2008). Identifying sources of intractability in cognitive models: An illustration using analogical structure mapping. In *Proceedings of the Annual Meeting of the Cognitive Science Society* (Vol. 30).
- Wolff, A., Berberian, N., Golesorkhi, M., Gomez-Pilar, J., Zilio, F., & Northoff, G. (2022). Intrinsic neural timescales: Temporal integration and segregation. *Trends in Cognitive Sciences*, 26(2), 159–173.
- Wyble, B., & Bowman, H. (2019). Temporal Segmentation for Faster and Better Learning. In *2019 Conference on Cognitive Computational Neuroscience*. Berlin, Germany: Cognitive Computational Neuroscience.
- Zacks, J. M., Braver, T. S., Sheridan, M., Donaldson, D. I., Snyder, a. Z., Ollinger, J. M., ... Raichle, M. E. (2001). Human brain activity time-locked to perceptual event boundaries. *Nature neuroscience*, 4(6), 651–655.