

Repetition Avoidance under Uniform Frequency Constraints: Limits of Sparse Perturbations

ALESSIO DRIVET
GeoGebra Institute of Turin
ITALY

Abstract: - We investigate the interaction between repetition avoidance and global distribution constraints in infinite words over a finite alphabet. We ask whether uniform letter frequencies can be imposed without increasing the classical repetition threshold.

To address this question, we introduce uniform variants of the repetition threshold and develop a general framework based on sparse perturbations. As a main structural result, we prove that insertions of vanishing density preserve asymptotic letter frequencies, showing that local corrections cannot enforce global uniformity. This identifies an inherent limitation of perturbative approaches.

We complement the theoretical analysis with computational experiments on prefixes up to 50 000. These experiments show that high uniformity and controlled repetition can coexist under relaxed thresholds for alphabets of size at least 3, whereas the binary case exhibits a distinct obstruction, with the maximum exponent exceeding the prescribed bound.

Taken together, our results suggest that the repetition threshold may be robust under uniformity constraints, while highlighting the fundamentally different roles of local and global methods. We conclude with a set of open problems toward explicit constructions combining optimal repetition bounds with uniform frequencies.

Key-Words: - combinatorics on words, repetition threshold, Dejean threshold, uniform letter frequencies, balanced words, repetition avoidance

Received: March 7, 2026. Revised: April 6, 2026. Accepted: May 17, 2026. Published: October 12, 2026.

1 Introduction

The study of repetitions in infinite words is a central topic in combinatorics on words, originating from the pioneering work of Axel Thue [1] and culminating in Dejean's theorem [2], [3], [4], which characterises the minimal exponent that cannot be avoided over a finite alphabet.

In parallel, global distribution properties of infinite words, such as uniform letter frequencies and balance conditions, play a fundamental role in symbolic dynamics and theoretical computer science. These properties impose long-range constraints on word structure and are typically studied independently of local combinatorial restrictions [5], [6].

This work is motivated by the observation that these two perspectives — local repetition avoidance and global distribution control — are not naturally aligned. Repetition constraints act at a local scale, forbidding highly periodic patterns in short factors, while uniformity is an asymptotic requirement that concerns the distribution of letters over arbitrarily long prefixes. Understanding how these two types of constraints interact is therefore a non-trivial and largely unexplored problem.

More precisely, we investigate whether it is possible to impose uniform letter frequencies while maintaining optimal or near-optimal bounds on repetitions. In particular, we ask whether the classical repetition threshold $RT(k)$ remains unchanged when one restricts attention to words with uniform asymptotic frequencies.

To formalise this question, we introduce constrained variants of the repetition threshold incorporating uniformity conditions. These parameters capture the minimal exponent above which repetitions cannot be avoided under global distribution constraints. While the classical repetition threshold and its finite-critical version are well understood [7], [8], [9], their behaviour under such additional constraints remains largely unknown. Related directions consist of repetition thresholds for binary rich words [10], avoidance of approximate repetitions measured by longest common subsequences [11], and abelian analogues based on letter counts [12].

Our approach is based on a general construction framework using sparse perturbations. Starting from an infinite word with controlled repetition properties, we introduce local corrections at increasingly distant

positions in order to enforce global distribution constraints. This strategy allows us to separate, to some extent, the local and global aspects of the problem.

As a first result, we show that sparse insertions of vanishing density preserve asymptotic letter frequencies (Lemma 5.1). This observation has an important conceptual consequence: sparse perturbations can be used to refine a construction without altering its limiting distribution, but they cannot create uniformity starting from a non-uniform base word.

Motivated by this limitation, we outline a research program aimed at constructing, *for every* $k \geq 2$ *and every* $\varepsilon > 0$, infinite words with uniform letter frequencies avoiding $(RT(k) + \varepsilon)^+ - \text{powers}$. We complement the theoretical framework with computational experiments on long finite prefixes, which illustrate both the limitations of local greedy constructions and the potential of non-local correction mechanisms.

The main contributions of this paper are as follows:

1. We introduce a framework for combining repetition avoidance with global distribution constraints via sparse perturbations.
2. We prove that sparse perturbations of vanishing density preserve asymptotic letter frequencies and identify the inherent limitations of this approach.
3. We provide computational evidence supporting the conjecture that the repetition threshold remains unchanged under uniformity constraints.
4. We formulate several open problems concerning uniform and balanced variants of the repetition threshold.

These results suggest that the classical repetition threshold may be robust under global distribution constraints, but a complete understanding of this phenomenon remains an open challenge.

2 Preliminaries

We begin by recalling the basic notions from combinatorics on words that will be used throughout the paper. Our goal is not only to fix notation, but also to highlight the quantities that will play a central role in the interaction between repetition constraints and global distribution properties [6].

Let A be a finite alphabet with size $k \geq 2$. An infinite word over A is a sequence $w = w_0 w_1 w_2 \dots$ where each w_i belongs to A .

A factor (or subword) of w is any finite word u of the form $u = w_i w_{i+1} \dots w_{i+\ell-1}$, for some integers $i \geq 0$ and $\ell \geq 1$.

A central notion in the study of repetition is periodicity. A positive integer p is called a period of a finite word $u = u_0 u_1 \dots u_{n-1}$ if $u_j = u_{j+p}$ for all indices j such that $0 \leq j < n - p$.

The exponent of a word u is defined as the ratio between its length and its minimal period. This quantity measures how many times a basic pattern is repeated inside u . For instance, squares correspond to exponent 2, cubes to exponent 3, and so on.

One of the central parameters in combinatorics on words is the repetition threshold $RT(k)$, introduced by Dejean [2]. It is defined as the infimum of all real numbers α such that there exists an infinite word over A avoiding all factors of exponent strictly greater than α . In other words, $RT(k)$ represents the optimal bound beyond which repetitions cannot be eliminated. The proof of Dejean's conjecture was developed through work on specific alphabet sizes [13], [14], [15] and on large alphabets [16], and was completed by Currie and Rampersad [3] and Rao [4].

In this work, we also consider a refined notion tracks how often critical repetitions occur. For a real number α , we say that an infinite word w avoids $\alpha^+ - \text{powers}$ if every factor u of w has exponent at most α . We say that w is α -finite-critical if it avoids $\alpha^+ - \text{powers}$ and contains only finitely many factors whose exponent is exactly $[\alpha]$, $[\alpha]$, $[\alpha]$.

We now turn to global distribution properties, which will be central in the rest of the paper. An infinite word w is said to have uniform letter frequencies if, for every letter a in A , the limit

$$f_a(w) = \lim_{N \rightarrow \infty} |w_0 \dots w_{N-1}|_a / N$$

exists and is equal to $1/k$. Informally, this means that each letter appears with the same asymptotic density along the word.

A related but stronger notion is that of balance. Given a constant $C \geq 0$, we say that a word w is C -balanced if, for all letters a and all factors u and v of equal length, the difference between the number of occurrences of a in u and in v is at most C [5], [6].

This condition requires that the distribution of letters is not only uniform in the limit, but also well controlled at every finite scale. The relationship between balance and critical exponent has been studied specifically for balanced sequences [17].

These notions — exponent, repetition threshold, uniform frequencies, and balance — capture the two types of constraints that we aim to combine: local control of repetitions and global control of letter distribution.

3 Definitions

Definition 3.1 (Uniform repetition threshold)

Let $k \geq 2$. We define $URT(k)$ as the infimum of all real numbers $\alpha \geq 1$ such that there exists an infinite word over a k -letter alphabet with uniform letter frequencies and avoiding α^+ -powers.

Definition 3.2 (Uniform finite-critical repetition threshold)

Let $k \geq 2$. We define $UFRT(k)$ as the infimum of all real numbers $\alpha \geq 1$ such that there exists an infinite word over a k -letter alphabet with uniform letter frequencies that is α -finite-critical; that is, it avoids α^+ -powers, and contains only finitely many factors of exponent exactly α .

Definition 3.3 (Finite-critical repetition threshold)

Let $k \geq 2$. We define $FRT(k)$ as the infimum of all real numbers $\alpha \geq 1$ such that there exists an infinite word over a k -letter alphabet that is α -finite-critical; that is, it avoids all α^+ -powers and contains only finitely many factors whose exponent is exactly α .

4 Conjectures

We investigate whether global distribution constraints increase the repetition threshold, and we formulate a corrected framework for this question under uniform frequency constraints.

Conjecture 4.1 (Uniform version).

For every $k \geq 2$,

$$URT(k) = RT(k).$$

Conjecture 4.2 (Balanced version).

For every $k \geq 2$,

$$UFRT(k) = FRT(k),$$

or at least

$$UFRT(k) = RT(k).$$

5 Sparse perturbations

We introduce a general framework for modifying infinite words by inserting sparse correction blocks, aiming to improve global distributional properties while preserving repetition constraints.

Lemma 5.1 (Preservation of frequencies under sparse perturbations).

Let A be a finite alphabet and let $w \in A^{\mathbb{N}}$ be an infinite word with well-defined letter frequencies. Let $(p_n)_{n \geq 1}$ be a strictly increasing sequence of positions and let $(b_n)_{n \geq 1}$ be a sequence of finite words over A ,

with $l_n = |b_n|$. Let w' be the word obtained from w by inserting each block b_n immediately after position p_n .

Assume that:

1) $\sum_{n: p_n \leq N} l_n = o(N)$ (vanishing total insertion density),

2) $l_n = o(p_n)$ (each insertion is locally negligible).

Then w' has the same asymptotic letter frequencies as w .

Proof.

The key idea is that the inserted blocks occupy only a negligible proportion of positions, so their contribution does not affect the limiting frequencies.

Fix a letter $a \in A$. For $M \geq 1$, consider the prefix $w'[0..M-1]$. Let $N = N(M)$ be the number of original symbols from w that appear in this prefix and let m be such that $p_m \leq N < p_{m+1}$.

We can write $M = N + L(N) + \varepsilon(M)$, where $L(N) = \sum_{n: p_n \leq N} l_n$ is the total length of inserted blocks fully contained in the prefix, and $\varepsilon(M)$ accounts for a possible partial inclusion of the next block.

By assumption, $L(N) = o(N)$, and $l_{m+1} = o(N)$. Therefore $M = N + o(N)$, which implies $N/M \rightarrow 1$ as $M \rightarrow \infty$.

We now decompose the number of occurrences of a in $w'[0..M-1]$: $w'[0..M-1]_a = w[0..M-1]_a + E_a(N)$,

where $E_a(N)$ is the contribution of all inserted blocks.

Since each block contributes at most l_n occurrences of a , we have $0 \leq E_a(N) \leq L(N) + l_{m+1}$, hence $E_a(N) = o(N)$.

Dividing by M , we obtain: $\frac{w'[0..M-1]_a}{M} = \left(\frac{w[0..N-1]_a}{N} \cdot \frac{N}{M} + \frac{E_a(N)}{M} \right)$.

As $M \rightarrow \infty$, the first factor tends to $f_a(w)$, the second to 1, and the third to 0. Therefore $|w'[0..M-1]_a|/M \rightarrow f_a(w)$.

Since a was arbitrary, w' has the same asymptotic letter frequencies as w . ■

The following statement is proposed as a plausible intermediate step toward the uniformization problem. While it is consistent with the framework developed above, a complete proof is currently not available and remains an open problem.

Open problem 5.2 (Approximate uniformization via sparse perturbations).

Let A be a finite alphabet with size $k \geq 2$, and let w be an infinite word over A with well-defined letter frequencies $f_a(w)$, assumed to be rational numbers.

Then, for every $\delta > 0$ and every $\varepsilon > 0$, there exists an infinite word w' over A such that:

1. w' avoids all factors whose exponent is strictly greater than $\alpha + \varepsilon$;
2. for every letter $a \in A$, the frequency of a in w' is close to uniform, in the sense that the difference between $f_a(w')$ and $1/k$ is at most δ ;
3. the word w' differs from w only on a set of positions whose asymptotic density is at most δ .

This open problem suggests that allowing a small but positive density of modifications may be sufficient to approximate a uniform distribution and provides a possible bridge between sparse perturbation techniques and block-based constructions (see Section 7).

While exact uniformization appears to require modifications of positive density in the infinite setting, a weaker form of uniformization can already be achieved on finite prefixes, as shown below.

Proposition 5.3 (Finite-prefix uniformization under sparse perturbations).

Let A be a finite alphabet with size $k \geq 2$, and let w be an infinite word over A .

Then, for every integer $N \geq 1$ and every real number $\delta > 0$, there exists a finite word u , obtained from the prefix of length N of w by modifying at most δN positions, such that the distribution of letters in u is approximately uniform.

More precisely, for every letter $a \in A$, the proportion of occurrences of a in u differs from $1/k$ by at most δ .

Proof (sketch).

Consider the prefix of length N of the word w and count how many times each letter appears. Let N/k be the ideal number of occurrences for each letter in a perfectly uniform distribution.

If a letter appears more than N/k times, some of its occurrences can be replaced by letters that appear less frequently. By redistributing letters in this way, one can balance the counts so that all letters occur with nearly the same frequency.

The total number of modifications needed is proportional to the initial imbalance and can be bounded by a fraction of N .

This result shows that approximate uniformity is always achievable at the finite level, suggesting that the main difficulty lies in extending such constructions to infinite words while controlling repetition constraints.

6 Examples

Example 6.1 (Thue–Morse word, $k = 2$).

The Thue–Morse word $t = 0110100110010110\dots$ is defined as the fixed point of the morphism $0 \mapsto 01, 1 \mapsto 10$. It is well known that t is overlap-free, hence avoids all factors of exponent greater than 2, and has uniform letter frequencies [18] $f_0(t) = f_1(t) = 1/2$. However, t contains infinitely many factors of exponent exactly 2 (for instance, the factor 0110 has period 2 and exponent 2, and contains infinitely many square factors of unbounded length). Therefore t is not 2-finite-critical, and the question of constructing a strictly 2-finite-critical binary word with uniform frequencies remains open. Related binary constructions with only finitely many distinct square factors were studied by Entringer, Jackson, and Schatz [19] and by Badkobeh and Crochemore [8].

Example 6.2 (Square-free words over three letters, $k = 3$).

It is known that there exist square-free infinite words over a three-letter alphabet, i.e., words avoiding all factors of exponent ≥ 2 [1]. Since $RT(3) = 7/4$ [2], square-free words satisfy a strictly stronger constraint than required by the repetition threshold.

However, naively constructed square-free words (e.g., via greedy extension) do not in general have uniform letter frequencies: the three letters may appear with unequal asymptotic densities, or the frequencies may not even converge. This illustrates that repetition avoidance and uniform distribution are independent constraints that must be jointly enforced.

Example 6.3 (Sparse perturbation of a Dejean word, $k = 4$).

For $k = 4$, the repetition threshold is $RT(4) = 7/5$ [13]. Consider an infinite word w over $\{0, 1, 2, 3\}$ that avoids $(7/5)^+$ -powers, constructed via a known morphic method [9], [13]. In general, w need not have uniform letter frequencies.

Applying the sparse perturbation scheme of Section 5 — e.g., inserting correction blocks at positions $p_n = 2^n$ — one can improve the uniformity of long finite prefixes while preserving repetition constraints locally. However, by Lemma 5.1, such perturbations do not change the asymptotic letter frequencies of the word. This illustrates both the usefulness and the limitations of the method.

Example 6.4 (Sturmian words and the balance–repetition trade-off).

Sturmian words are 1-balanced [6] and have uniform letter frequencies by construction, but their critical exponent is strictly greater than $RT(2) = 2$: for instance, the Fibonacci word has critical exponent $(5+\sqrt{5})/2 \approx 3.618$ [6]. This shows that balance and uniform distribution alone do not force optimal repetition bounds, and that the interaction studied in this paper is genuinely non-trivial. In particular, Sturmian words do not witness Conjecture 4.2 for $k = 2$.

7 Main objective and open directions

The limitations of sparse perturbations highlighted in Section 5, together with Open Problem 5.2, suggest that achieving uniform frequencies may require constructions that balance local corrections with global structure.

The main goal of this work is to understand whether global distribution constraints can be reconciled with optimal bounds on repetitions.

More precisely, given an alphabet of size $k \geq 2$ and a parameter $\varepsilon > 0$, we seek to construct an infinite word with uniform letter frequencies that avoids all factors of exponent greater than $RT(k) + \varepsilon$. Establishing the existence of such words would show that the imposition of uniformity does not increase the repetition threshold.

At a conceptual level, the difficulty of this problem lies in the interaction between two fundamentally different types of constraints. Repetition avoidance is a local condition, while uniform letter frequencies are a global condition. These requirements are not obviously compatible, and their coexistence imposes non-trivial architectural limitations on the word.

The sparse perturbation framework developed in this paper provides a first step toward addressing this tension. Lemma 5.1 shows that insertions of vanishing density preserve asymptotic frequencies but also underlines a limitation: such perturbations cannot alter the limiting frequencies of a word.

This observation suggests that any successful construction must follow one of two broad strategies. The first approach is to start from a base word that already satisfies both constraints. The second approach is to design a genuinely global construction in which uniformity is enforced throughout the process.

The computational experiments presented in Section 8 provide evidence that these strategies are plausible, at least for long finite prefixes. However, extending these observations to infinite words remains an important challenge.

A natural intermediate step is to understand whether finite constructions with near-uniform distribution and controlled repetition can be extended indefinitely.

Heuristic 7.1 (Block construction).

For every $k \geq 2$, every $\alpha > RT(k)$, and every integer $L \geq 1$, there should exist finite words b such that:

- (i) $|b| = O(L)$;
- (ii) b avoids α^+ -powers;
- (iii) the letter frequencies in b differ from $1/k$ by at most $O(1/L)$.

These questions illustrate that the interaction between repetition avoidance and global distribution constraints remains poorly understood and suggest several directions for future research.

8 Computational Experiments

The goal of this section is to provide an exploratory computational study of the interaction between repetition avoidance and global distribution constraints. These experiments are not intended as proofs, but rather as qualitative evidence highlighting structural phenomena.

We consider finite prefixes of length up to $n = 50,000$, with exponent-related quantities evaluated on factors of bounded length (window size $W = 200$).

8.1 Methodology

We consider three construction strategies:

1. Thue–Morse (baseline, $k = 2$). The Thue–Morse word is used as a reference example in the binary case, as it is known to be overlap-free and to have uniform letter frequencies [18].
2. Greedy constrained construction (strict version). Starting from the empty word, at each step we append the least frequent letter that does not create a factor whose exponent exceeds a prescribed threshold α . If no such letter exists, the construction stops (deadlock). This version does not force any invalid extension.
3. Sparse perturbation (heuristic). Starting from a base word satisfying a relaxed constraint (threshold $\alpha + \delta$), we attempt to insert correction letters at exponentially increasing positions in order to improve uniformity. Insertions that would violate the tested threshold are skipped.

For each generated word, we measure: $\text{MaxExp}_{\leq W}$ (the maximum exponent among factors of length at most W); Unif (a prefix uniformity score

in $[0,1]$, equal to 1 for exact uniform distribution); $\#Crit \leq W$ (the number of distinct factors of length at most W whose exponent equals the tested threshold); and C_{prefix} (the maximum imbalance over all equal-length factors of the prefix). All exponent-related quantities are computed only up to the fixed window size $W = 200$.

8.2 Greedy horizon and local limitations

We first study the behaviour of the greedy construction as a function of the threshold. The results are summarised in Table 1.

Table 1. Greedy deadlock horizon under relaxed thresholds

k	tested a	built n	deadlocks	completed	MaxE xp	Unif
2	2.0000	24	1	0	2.0000	1.0000
2	2.0200	24	1	0	2.0000	1.0000
2	2.0500	24	1	0	2.0000	1.0000
2	2.1000	24	1	0	2.0000	1.0000
2	2.2000	13	1	0	2.1667	0.9231
2	2.3000	13	1	0	2.1667	0.9231
3	1.7500	58	1	0	1.7500	0.9770
3	1.7700	37	1	0	1.7619	0.9640
3	1.8000	37	1	0	1.7619	0.9640
3	1.8500	45	1	0	1.8333	1.0000
3	1.9500	40	1	0	1.9048	0.9667
3	2.0500	50000	0	1	2.0417	1.0000
4	1.4000	12	1	0	1.3333	0.8333
4	1.4200	12	1	0	1.3333	0.8333
4	1.4500	18	1	0	1.4444	0.8889
4	1.5000	50000	0	1	1.5000	1.0000
4	1.6000	50000	0	1	1.5962	1.0000
4	1.7000	50001	0	1	1.6667	1.0000

The data in Table 1 show that the greedy constrained construction systematically encounters a finite deadlock horizon at the exact repetition

threshold, independently of the target length n . This confirms that the obstruction is local in nature and not due to insufficient prefix length.

Relaxing the threshold increases the achievable length but does not eliminate the phenomenon entirely, indicating that purely local strategies are inherently limited in enforcing repetition constraints alongside global distribution requirements.

8.3 Long-prefix behaviour under relaxed thresholds

We next investigate the behaviour of sparse perturbation constructions under relaxed thresholds $\alpha + \delta$, focusing on prefixes of length up to $n = 50,000$. The results are reported in Table 2.

Table 2. Long-prefix study

Method	tested a	Built n	MaxE xp	Unif	#Cr it	C_p ref
Thue-Morse	2.0000	50000	2.0000	1.0000	26	2
Greedy (strict)	2.0000	24	2.0000	1.0000	9	2
Sparse perturbation	2.3000	50000	3.0000	1.0000	0	3
Greedy (strict)	1.7500	58	1.7500	0.9770	3	3
Sparse perturbation	2.0500	50012	2.0417	1.0000	0	4
Greedy (strict)	1.4000	12	1.3333	0.8333	0	1
Sparse perturbation	1.7000	50000	1.6667	1.0000	0	3

Table 2 shows that high uniformity and controlled repetition can coexist on long prefixes under relaxed thresholds, particularly for $k \geq 3$, where the sparse perturbation method achieves near-perfect uniformity while keeping the maximum exponent below the prescribed bound.

By contrast, for $k = 2$, the method fails to control the exponent, which exceeds the relaxed threshold as the prefix length increases. This highlights a structural asymmetry between the binary and larger alphabets and suggests that the main difficulty of the problem lies in reconciling global uniformity with local repetition constraints in the most constrained case.

Overall, these experiments suggest that while global uniformity and repetition control can coexist on long finite prefixes, their compatibility depends strongly on the alphabet size and cannot be achieved by purely local methods.

9 Open problems

The results presented in this work provide evidence for Conjecture 4.1 but leave several natural questions open.

1. Uniform repetition threshold.
Does there exist, for every $k \geq 2$ and every $\varepsilon > 0$, an infinite word over a k -letter alphabet with uniform letter frequencies avoiding $(RT(k) + \varepsilon)^+$ -powers?
2. Finite-critical uniform version.
Does there exist, for every $k \geq 2$, an infinite word with uniform frequencies that is $RT(k)$ finite-critical?
3. Balanced constructions.
Determine whether the balanced variant satisfies $UFRT_{bal}(k) = RT(k)$.
4. Explicit constructions.
Can one give an explicit and efficiently computable construction achieving both uniform frequencies and near-optimal repetition bounds?
5. Extension from finite prefixes.
Under what conditions can long finite prefixes with near-uniform distribution and low repetition be extended to infinite words?
6. Limits of local methods.
To what extent can local control of repetitions (within bounded windows) be extended to global guarantees?
7. Behaviour of sparse perturbations for $k = 2$.
Computational experiments show that, for $k = 2$, the maximum exponent produced by the sparse perturbation method grows with the prefix length (reaching 3.0 at $n = 50,000$), rather than stabilising below the relaxed threshold. Does there exist a sparse correction scheme for $k = 2$ that controls the maximum exponent asymptotically, or is this failure an intrinsic obstruction of the binary alphabet?
8. Existence of near-uniform blocks.
For every $k \geq 2$, every, $\alpha > RT(k)$ and every $L \geq 1$, do there exist finite words b over a k -letter alphabet such that: (i) $|b| = O(L)$; (ii) b avoids α^+ -powers; (iii) the letter frequencies in b differ from $1/k$ by at most $O(1/L)$? A positive answer would enable a block-construction approach to the uniform repetition threshold.

10 Conclusion

We introduced global distribution constraints into the study of repetition avoidance in infinite words, focusing in particular on uniform letter frequencies.

We proposed a sparse perturbation framework and proved that insertions of vanishing density preserve asymptotic letter frequencies. This highlights both the usefulness and the limitations of such perturbations: they allow local corrections without affecting global distribution but cannot create uniformity from a non-uniform base word.

Our computational experiments on prefixes of length up to $n = 50,000$ (with window size $W = 200$) show that low repetition and near-uniform distribution can coexist on long finite prefixes under relaxed thresholds for $k \geq 3$, and that this behaviour is stable over significantly longer words than previously tested. By contrast, for $k = 2$, the sparse perturbation method not only violates the relaxed bound but produces a maximum exponent that worsens with n (from 2.5 at $n = 5,000$ to 3.0 at $n = 50,000$), indicating a structural difficulty specific to the binary case. Purely local greedy constructions encounter intrinsic limitations at the exact threshold regardless of prefix length.

These observations support the conjecture that the uniform repetition threshold coincides with the classical repetition threshold: $URT(k) = RT(k)$. However, this conjecture remains open, as does its finite-critical counterpart. Establishing these results rigorously, as well as extending them to balanced words, remains an interesting direction for future research.

Acknowledgement:

The author acknowledges the use of AI-based tools for suggestions, language revision, and coding assistance.

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this work, the author(s) utilized artificial intelligence (AI) technologies solely for the purpose of linguistic refinement and grammatical correction. Following the application of these tools, the author(s) comprehensively reviewed, verified, and edited the resulting text. The author(s) maintain full accountability and responsibility for the accuracy, integrity, and final content of the publication.

References:

- [1] Berstel, J. (1995). Axel Thue's papers on repetitions in words: A translation

- (Publications du LaCIM No. 20). Université du Québec à Montréal.
- [2] Dejean, F. (1972). Sur un théorème de Thue. *Journal of Combinatorial Theory, Series A*, 13(1), 90–99.
 - [3] Currie, J. D., & Rampersad, N. (2011). A proof of Dejean's conjecture. *Mathematics of Computation*, 80(274), 1063–1070.
 - [4] Rao, M. (2011). Last cases of Dejean's conjecture. *Theoretical Computer Science*, 412(27), 3010–3018.
 - [5] Berthé, V., & Rigo, M. (Eds.). (2010). *Combinatorics, automata and number theory*. Cambridge University Press.
 - [6] Lothaire, M. (2002). *Algebraic combinatorics on words*. Cambridge University Press.
 - [7] Badkobeh, G., & Crochemore, M. (2011). Finite-repetition threshold for infinite ternary words. *Electronic Proceedings in Theoretical Computer Science*, 63, 37–43. <https://doi.org/10.4204/EPTCS.63.7>
 - [8] Badkobeh, G., & Crochemore, M. (2012). Fewest repetitions in infinite binary words. *RAIRO—Theoretical Informatics and Applications*, 46(1), 17–31. <https://doi.org/10.1051/ita/2011109>
 - [9] Badkobeh, G., Crochemore, M., & Rao, M. (2014). Finite repetition threshold for large alphabets. *RAIRO—Theoretical Informatics and Applications*, 48(4), 419–430.
 - [10] Currie, J. D., Mol, L., & Rampersad, N. (2020). The repetition threshold for binary rich words. *Discrete Mathematics & Theoretical Computer Science*, 22(1), Article 6. <https://doi.org/10.23638/DMTCS-22-1-6>
 - [11] Camungol, S., & Rampersad, N. (2016). Avoiding approximate repetitions with respect to the longest common subsequence distance. *Involve*, 9(4), 657–666.
 - [12] Fici, G., & Puzynina, S. (2023). Abelian combinatorics on words: A survey. *Computer Science Review*, 47, Article 100532.
 - [13] Pansiot, J.-J. (1984). On a conjecture of F. Dejean on repetitions in words. *Discrete Applied Mathematics*, 7(3), 297–311.
 - [14] Ollagnier, J.-M. (1992). Proof of Dejean's conjecture for alphabets with 5 to 11 letters. *Theoretical Computer Science*, 95(2), 187–205.
 - [15] Mohammad-Noori, M., & Currie, J. D. (2007). Dejean's conjecture and Sturmian words. *European Journal of Combinatorics*, 28(3), 876–890.
 - [16] Carpi, A. (2007). On Dejean's conjecture over large alphabets. *Theoretical Computer Science*, 385(1–3), 137–151.
 - [17] Dvořáková, L., Pelantová, E., Opočenská, D., & Shur, A. M. (2022). On minimal critical exponent of balanced sequences. *Theoretical Computer Science*, 922, 158–169.
 - [18] Allouche, J.-P., & Shallit, J. (2003). *Automatic sequences: Theory, applications, generalizations*. Cambridge University Press.
 - [19] Entringer, R. C., Jackson, D. E., & Schatz, J. A. (1974). On nonrepetitive sequences. *Journal of Combinatorial Theory, Series A*, 16(2), 159–164.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The author contributed to the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The author has no conflicts of interest to declare that are relevant to the content of this article.

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)

This article is published under the terms of the Creative Commons Attribution License 4.0 https://creativecommons.org/licenses/by/4.0/deed.en_US

APPENDIX: PYTHON CODE

```
from __future__ import annotations
from collections import Counter
from dataclasses import dataclass
from typing import Dict, List, Optional

import numpy as np
# -----
# Repetition threshold
# -----
def RT(k: int) -> float:
    """Return the classical repetition threshold RT(k)."""
    if k < 2:
        raise ValueError("Alphabet size k must be at least 2.")
    if k == 2: return 2.0
    if k == 3: return 7 / 4
    if k == 4: return 7 / 5
    return k / (k - 1)
# -----
# Z-array and minimal period (O(n))
# -----
def _z_array(s: str) -> List[int]:
    n = len(s)
    z = [0] * n
    if n == 0:
        return z
    z[0] = n
    l = r = 0
    for i in range(1, n):
        if i < r:
            z[i] = min(r - i, z[i - l])
            while i + z[i] < n and s[z[i]] == s[i + z[i]]:
                z[i] += 1
        if i + z[i] > r:
            l, r = i, i + z[i]
    return z
def _minimal_period(s: str) -> int:
    n = len(s)
    if n <= 1:
        return n
    z = _z_array(s)
    for p in range(1, n):
        if z[p] >= n - p:
            return p
    return n
# -----
# Period scan for max exponent (O(n · W / max_exp))
# -----
```

```
def _max_exponent_period_scan(word: str, max_window: int) -> float:
    n = len(word)
    if n == 0:
        return 1.0
    max_exp = 1.0
    for p in range(1, max_window):
        if max_window / p <= max_exp:
            break
    consec = 0
    for i in range(p, n):
        if word[i] == word[i - p]:
            consec += 1
        else:
            consec = 0
    run_len = consec + p
    if run_len > max_window:
        run_len = (max_window // p) * p
    exp = run_len / p
    if exp > max_exp:
        max_exp = exp
    return max_exp

# -----
# Incremental suffix checker (used inside greedy construction)
# -----

class IncrementalSuffixChecker:
    def __init__(self, threshold: float, max_window: int) -> None:
        self.threshold = threshold
        self.max_window = max_window
        self.max_p: int = int(max_window / threshold) + 1
        self.word: List[str] = []
        self.consec: List[int] = [0] * (self.max_p + 2)
    def _update(self, char: str) -> bool:
        self.word.append(char)
        n = len(self.word)
        for p in range(1, min(n - 1, self.max_p) + 1):
            if self.word[n - 1] == self.word[n - 1 - p]:
                self.consec[p] += 1
            else:
                self.consec[p] = 0
        run_len = self.consec[p] + p
        if run_len > self.max_window:
            run_len = (self.max_window // p) * p
        if run_len / p > self.threshold:
            return False
        return True
    def try_append(self, char: str) -> bool:
        saved = self.consec[:]
        ok = self._update(char)
        if not ok:
            self.word.pop()
            self.consec = saved
        return ok
    def force_append(self, char: str) -> None:
```

```
self._update(char)
def current_word(self) -> str:
return "".join(self.word)
# -----
# Local analysis utilities
# -----
class LocalAnalyzer:
    def __init__(self, k: int, max_window: int = 200) -> None:
if k < 2:
    raise ValueError("Alphabet size k must be at least 2.")
if max_window < 1:
    raise ValueError("max_window must be at least 1.")
self.k = k
self.alphabet: List[str] = [str(i) for i in range(k)]
self.max_window = max_window
    @staticmethod
    def minimal_period(factor: str) -> int:
return _minimal_period(factor)
    def exponent(self, factor: str) -> float:
if not factor:
return 1.0
return len(factor) / _minimal_period(factor)
    def suffix_respects_threshold(self, word: List[str], threshold: float) -> bool:
n = len(word)
limit = min(n, self.max_window)
joined = "".join(word)
for length in range(2, limit + 1):
    sub = joined[n - length:]
    max_p = min(length - 1, int(length / threshold))
if max_p < 1:
continue
z = _z_array(sub)
for p in range(1, max_p + 1):
if z[p] >= length - p:
if length / p > threshold:
return False
break
return True
    def insertion_respects_threshold(
        self, word: str, insert_pos: int, letter: str, threshold: float,
    ) -> bool:
        candidate = word[:insert_pos] + letter + word[insert_pos:]
n = len(candidate)
limit = min(n, self.max_window)
for length in range(2, limit + 1):
start_lo = max(0, insert_pos - length + 1)
start_hi = min(insert_pos, n - length) + 1
max_p = min(length - 1, int(length / threshold))
if max_p < 1:
continue
for start in range(start_lo, start_hi):
sub = candidate[start: start + length]
z = _z_array(sub)
for p in range(1, max_p + 1):
if z[p] >= length - p:
```

```
if length / p > threshold:
    return False
break
return True
    def max_exponent_up_to_window(self, word: str) -> float:
        return max_exponent_period_scan(word, self.max_window)
def count_critical_factors_up_to_window(
    self, word: str, threshold: float, tol: float = 1e-9
) -> int:
    n = len(word)
    critical: set[str] = set()
    max_p = min(n - 1, int(self.max_window / threshold) + 1)
    for p in range(1, max_p + 1):
        target_L = p * threshold
        # Check all nearby integer lengths that satisfy |L/p - threshold| < tol
        for L in sorted({int(target_L), round(int(target_L)), int(target_L) + 1}):
            if L <= p or L > self.max_window or L > n:
                continue
            if abs(L / p - threshold) >= tol:
                continue
            # Check all windows of length L for period exactly p
            for i in range(n - L + 1):
                sub = word[i: i + L]
                # First check: does sub have period p?
                if any(sub[j] != sub[j - p] for j in range(p, L)):
                    continue
                # Confirm p is the MINIMAL period (no smaller period works)
            is_minimal = not any(
                all(sub[j] == sub[j - q] for j in range(q, L))
                for q in range(1, p)
            )
            if is_minimal:
                critical.add(sub)
    return len(critical)
    def prefix_uniformity_score(self, word: str) -> float:
        n = len(word)
        if n == 0:
            return 0.0
        counts = Counter(word)
        ideal = 1.0 / self.k
        tvd = sum(abs(counts.get(c, 0) / n - ideal) for c in self.alphabet) / 2
        return max(0.0, 1.0 - 2.0 * tvd)
    def prefix_balance_constant(self, word: str, patience: int = 500) -> int:
        n = len(word)
        if n == 0:
            return 0
        arr = np.array([int(c) for c in word], dtype=np.int32)
        max_imbalance = 0
        for a in self.alphabet:
            ind = (arr == int(a)).astype(np.int32)
            prefix = np.concatenate([0, np.cumsum(ind)])
            letter_max = 0
            no_improve = 0
            for length in range(1, n + 1):
                diffs = prefix[length:] - prefix[:n - length + 1]
```

```
imb = int(diffs.max() - diffs.min())
if imb > letter_max:
    letter_max = imb
no_improve = 0
else:
    no_improve += 1
if no_improve >= patience:
    break
if letter_max > max_imbalance:
    max_imbalance = letter_max
return max_imbalance
# -----
# Result dataclasses
# -----
@dataclass
class GreedyResult:
    word: str
    deadlock_count: int
    completed: bool
    stopped_at_length: int
@dataclass
class SparsePerturbationResult:
    word: str
    corrections: int
    skipped_insertions: int
    attempted_insertions: int
# -----
# Word generators
# -----
class WordGenerator:
    @staticmethod
    def thue_morse(length: int) -> str:
        return "".join(str(bin(n).count("1") % 2) for n in range(length))
    @staticmethod
    def greedy_constrained(
        k: int, length: int, threshold: float,
        analyzer: LocalAnalyzer, force_on_deadlock: bool = False,
    ) -> GreedyResult:
        alphabet = [str(i) for i in range(k)]
        checker = IncrementalSuffixChecker(threshold, analyzer.max_window)
        counts: Dict[str, int] = {c: 0 for c in alphabet}
        deadlocks = 0
        for _ in range(length):
            candidates = sorted(alphabet, key=lambda c: (counts[c], c))
            chosen: Optional[str] = None
            for char in candidates:
                if checker.try_append(char):
                    chosen = char
                    break
            if chosen is not None:
                counts[chosen] += 1
            else:
                deadlocks += 1
            if force_on_deadlock:
                forced = candidates[0]
```

```

checker.force_append(forced)
counts[forced] += 1
else:
w = checker.current_word()
    return GreedyResult(w, deadlocks, False, len(w))
w = checker.current_word()
    return GreedyResult(w, deadlocks, True, len(w))
@staticmethod
def sparse_perturbation(
    base_word: str, k: int, threshold: float, analyzer: LocalAnalyzer,
) -> SparsePerturbationResult:
    alphabet = [str(i) for i in range(k)]
    word_list: List[str] = list(base_word)
    corrections = skipped = attempted = 0
    exp_step = 1
    while True:
        index = 2 ** exp_step
        if index >= len(word_list):
            break
        attempted += 1
        counts = Counter(word_list[:index])
        least_freq = min(alphabet, key=lambda c: (counts[c], c))
        word_str = "".join(word_list)
        if analyzer.insertion_respects_threshold(word_str, index, least_freq, threshold):
            word_list.insert(index, least_freq)
            corrections += 1
        else:
            skipped += 1
            exp_step += 1
        return SparsePerturbationResult("".join(word_list), corrections, skipped, attempted)
# -----
# Experiment runners
# -----
def run_table1(max_window: int = 200) -> None:
    """Greedy deadlock horizon scan (Table 1)."""
    configs = [
        (2, [2.00, 2.02, 2.05, 2.10, 2.20, 2.30]),
        (3, [1.75, 1.77, 1.80, 1.85, 1.95, 2.05]),
        (4, [1.40, 1.42, 1.45, 1.50, 1.60, 1.70]),
    ]
    target_n = 50000
    print(f'n={target_n:,} W={max_window}')
    print(f'{k:>3} {alpha:>8} {built_n:>8} {deadlocks:>10} '
          f'{completed:>10} {MaxExp:>8} {Unif:>8}')
    print("-" * 72)
    for k, alphas in configs:
        analyzer = LocalAnalyzer(k, max_window)
        for alpha in alphas:
            result = WordGenerator.greedy_constrained(k, target_n, alpha, analyzer)
            w = result.word
            max_exp = analyzer.max_exponent_up_to_window(w) if w else 0.0
            unif = analyzer.prefix_uniformity_score(w) if w else 0.0
            print(f'{k:>3} {alpha:>8.4f} {result.stopped_at_length:>8} '
                  f'{result.deadlock_count:>10} {int(result.completed):>10} '
                  f'{max_exp:>8.4f} {unif:>8.4f}')
    def run_table2(max_window: int = 200, target_n: int = 50_000) -> None:

```

```
configs = [(2, 2.00, 2.30), (3, 1.75, 2.05), (4, 1.40, 1.70)]
print(f'n={target_n:,} W={max_window}')
print(f'{'method':22} {'k':>2} {'alpha':>6} {'n_built':>8} "
      f'{'MaxExp':>8} {'Unif':>6} {'#Crit':>6} {'C_pref':>6}")
print("-" * 82)
for k, strict_alpha, relaxed_alpha in configs:
a_s = LocalAnalyzer(k, max_window)
a_r = LocalAnalyzer(k, max_window)
if k == 2:
tm = WordGenerator.thue_morse(target_n)
print(f'{'Thue-Morse':22} {'k':>2} {2.0:>6.4f} {len(tm):>8} "
      f'{'a_s.max_exponent_up_to_window(tm):>8.4f} "
f'{'a_s.prefix_uniformity_score(tm):>6.4f} "
      f'{'a_s.count_critical_factors_up_to_window(tm, 2.0):>6} "
f'{'a_s.prefix_balance_constant(tm):>6}")
g = WordGenerator.greedy_constrained(k, target_n, strict_alpha, a_s)
w = g.word
print(f'{'Greedy (strict)':22} {'k':>2} {strict_alpha:>6.4f} {len(w):>8} "
      f'{'a_s.max_exponent_up_to_window(w) if w else 0:>8.4f} "
      f'{'a_s.prefix_uniformity_score(w) if w else 0:>6.4f} "
      f'{'a_s.count_critical_factors_up_to_window(w, strict_alpha) if w else 0:>6} "
      f'{'a_s.prefix_balance_constant(w) if w else 0:>6}")
base = WordGenerator.greedy_constrained(
    k, target_n, relaxed_alpha, a_r, force_on_deadlock=True
).word
sp = WordGenerator.sparse_perturbation(base, k, relaxed_alpha, a_r)
w2 = sp.word
print(f'{'Sparse perturbation':22} {'k':>2} {relaxed_alpha:>6.4f} {len(w2):>8} "
      f'{'a_r.max_exponent_up_to_window(w2):>8.4f} "
f'{'a_r.prefix_uniformity_score(w2):>6.4f} "
      f'{'a_r.count_critical_factors_up_to_window(w2, relaxed_alpha):>6} "
f'{'a_r.prefix_balance_constant(w2):>6}")
if __name__ == "__main__":
    print("=== Table 1: Greedy deadlock horizon ===\n")
run_table1()
    print("\n=== Table 2: Long-prefix study ===\n")
run_table2()
```