Bounding the Fragmentation of B-Trees Subject to Batched Insertions

MICHAEL A. BENDER, Stony Brook University and RelationalAI, USA
AARON BERNSTEIN, New York University, USA
NAIREN CAO, New York University, USA
ALEX CONWAY, Cornell Tech, USA
MARTÍN FARACH-COLTON, New York University, USA
HANNA KOMLÓS, Max Planck Institute for Informatics, Germany
YARIN SHECHTER, New York University, USA
NICOLE WEIN, University of Michigan, USA

The issue of internal fragmentation in data structures is a fundamental challenge in database design. A seminal result of Yao in this field shows that evenly splitting the leaves of a B-tree against a workload of uniformly random insertions achieves space utilization of around 69%. However, many database applications perform batched insertions, where a small run of consecutive keys is inserted at a single position. We develop a generalization of Yao's analysis to provide rigorous treatment of such batched workloads. Our approach revisits and reformulates the analytical structure underlying Yao's result in a way that enables generalization and is used to argue that even splitting works well for many workloads in our extended class. For the remaining workloads, we develop simple alternative strategies that provably maintain good space utilization.

CCS Concepts: • **Theory of computation → Data structures design and analysis**; *Predecessor queries*; *Sorting and searching*; **Online algorithms**; *Randomness, geometry and discrete structures*; *Database theory*; **Data structures and algorithms for data management**.

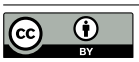
Additional Key Words and Phrases: Data structures, Database design, B-trees, External memory, Internal fragmentation, Memory fragmentation, Space utilization, Theory

ACM Reference Format:
Michael A. Bender, Aaron Bernstein, Nairen Cao, Alex Conway, Martín Farach-Colton, Hanna Komlós, Yarin Shechter, and Nicole Wein. 2026. Bounding the Fragmentation of B-Trees Subject to Batched Insertions. *Proc. ACM Manag. Data* 4, 2, Article 098 (May 2026), 25 pages. https://doi.org/10.1145/3801894

1 Introduction

The issue of internal fragmentation in data structures is a fundamental challenge in database design [6, 12, 15, 22]. The issue is that when a leaf of a B-tree fills, it must be split (evenly or otherwise), and the average fill of the resulting leaves is 50%, no matter how the leaves are split. Underfull leaves cause a number of well-studied performance degradations, including more cache misses [17],

Authors' Contact Information: Michael A. Bender, Stony Brook University and RelationalAI, Stony Brook, NY, USA, bender@cs.stonybrook.edu; Aaron Bernstein, New York University, New York, NY, USA, aaron.bernstein@nyu.edu; Nairen Cao, New York University, New York, NY, USA, nc1827@nyu.edu; Alex Conway, Cornell Tech, New York, NY, USA, me@ajhconway.com; Martín Farach-Colton, New York University, New York, NY, USA, martin@farach-colton.com; Hanna Komlós, Max Planck Institute for Informatics, Saarbrücken, Germany, hkomlos@gmail.com; Yarin Shechter, New York University, New York, NY, USA, yahrin4@gmail.com; Nicole Wein, University of Michigan, Ann Arbor, MI, USA, nswein@umich.edu.



ACM 2836-6573/2026/5-ART098
https://doi.org/10.1145/3801894

space management on NVMe [5], computational slowdown in in-memory databases [18], tree rebalancing costs [23], etc.

Studying the fill of the leaves is called *fringe analysis* [7, 11], and for B-trees, the fill of the leaves is the high-order term in the overall fill because all but a $\Theta(1/B)$ fraction of nodes are leaves. In this paper, we study the fill of B-tree leaves, specifically, we define the ***block-splitting problem*** as follows. Insertions arrive online, where each insertion is at a specified ***rank*** (between two previously inserted elements, or at the beginning or end). The algorithm must maintain a partition of the elements into ***blocks*** of size at most B each, where each block is a contiguous set of elements. That is, once a block reaches size $B + 1$, the algorithm must irrevocably split it into two blocks.[1] The ***fill*** achieved by an algorithm (after sufficiently many insertions) is the average over all blocks of the number of elements in the block divided by B.

A critical special case for the block-splitting problem is when insertions arrive one at a time at uniformly random ranks. Splitting blocks evenly results in a distributions of fills from 50% full to completely full. If this distribution were even, the average fill of a leaf would be 75%, but fuller blocks are more likely to receive insertions than emptier blocks. Thus, the true answer should be less than 75% but greater than 50%.

A seminal result of Yao from 1978 [21, 25] shows that evenly splitting the leaves of a B-tree under uniform insertions results in an average fill of $\ln 2 \approx 69\%$. This model of insertions is quite common in real databases, as reported by Facebook [9], InnoDB [10], Oracle [2], and Microsoft SQL Server [3].

For many databases, however, uniformly random insertions do not accurately model typical usage; instead, databases often witness ***batches*** of sequential insertions [4, 8, 16, 24, 26]. In this case, an insertion operation involves inserting r items, one after another, sequentially in key space. Many databases and key-value stores have specific heuristics for increasing the fill of leaves in the case of batch insertions [1, 13, 19].

In this paper, we generalize Yao's approach to the batch insertion case: with a fixed batch size r, a uniformly random rank is (repeatedly) selected and r sequential insertions are performed at that rank. It might seem that as r grows, the problem of achieving high fill would get easier. After all, we known more information about the insertion pattern into the database as more items are inserted in each location. However, we find that as least for even splitting and the closely related ***deferred even splitting***, which we define below, the picture isn't so clear. Before we explore the complexity of the varying r, we describe a few splitting algorithms:

- **Even splitting.** We process the r new elements one by one. Whenever a block grows from size B to size $B + 1$, we split it into two blocks of sizes $\lfloor (B + 1)/2 \rfloor$ and $\lceil (B + 1)/2 \rceil$.
- **Deferred even splitting.** Suppose r elements are inserted into a block that starts out with with ℓ elements, and that $r + \ell > B$. The deferred even split strategy replaces this block with the minimum number of blocks possible so that all resulting blocks differ in fill by at most 1. That is, we break the block into $\lceil (r + \ell)/B \rceil$ blocks with the same fill, up to rounding.[2]

Before stating our results, we demonstrate the surprisingly erratic behavior of these simple algorithms for various values of the batch size r with data from simulations. Figure 1 shows the measured fill achieved by even splitting and deferred even splitting when $r < B$. In the latter case,

[1] Beyond the choice of splitting rule, one can also mitigate internal fragmentation using more global mechanisms, such as rebuilding the tree or allowing keys to migrate across block boundaries. However, these approaches are typically much more expensive than the simple local splitting strategies studied in this paper. Thus, the more fragmentation can be mitigated by such simple strategies, the less often one must rely on these heavier mechanisms, reducing the overall cost of storage management.

[2] The splitting in this strategy is not literally deferred. It can be implemented so that the splitting happens as items get inserted, rather than requiring a block to get overfull before splitting.

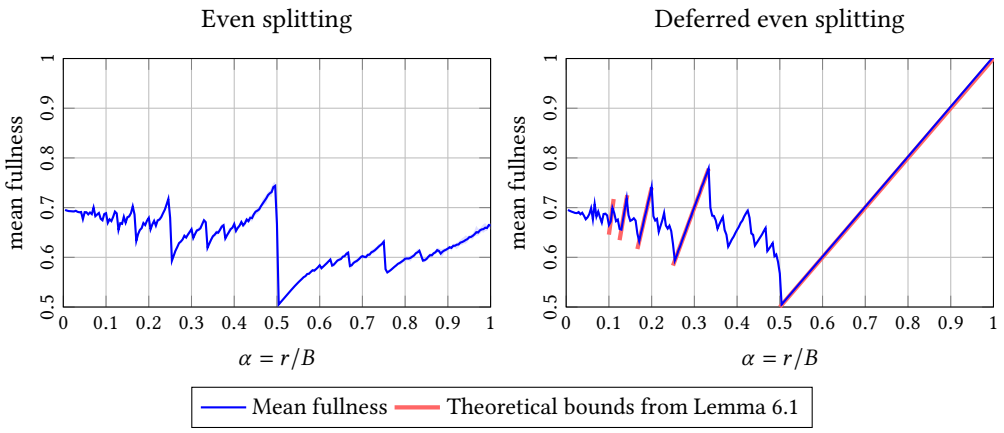


Fig. 1. Experimental fullness of deferred and even splitting on batch insertions on blocks of size $B = 240$. 200,000 insertions are made from empty using batch insertions of varying length ($r \in [1, B]$). The mean fullness across 10 independent runs is shown. Bounds from Lemma 6.1 are shown in red for comparison.

we also prove tight bounds on the fill in Lemma 6.1, and we see that these bounds align with our experiments.

We observe two interesting phenomena in Figure 1. The first is that the average fill varies quite unpredictably with r. Nonetheless, even splitting performs well in our simulations, as long as r is relatively small. The second observation is that, unfortunately, according to the simulations, these algorithms do not always get better than the trivial 50% fill. In particular, for $r = B/2$, both algorithms get exactly 50% fill. However, we show that this undesirable behavior is not inherent to batch insertions, and interestingly, we can achieve better fill by employing *uneven* splitting.

- **(Deferred) Uneven splitting.** Suppose we are inserting r elements into a block that starts out with ℓ elements, and that $r + \ell \in (B, 2B)$. We split the block into two blocks of sizes $\lfloor \delta(r + \ell)/2 \rfloor$ and $r + \ell - \lfloor \delta(\ell + r)/2 \rfloor$, where $\delta \in (0, 1)$ is the ***splitting factor*** for this split.[3] Note that this rule is only defined when both resulting blocks have at most B items.

In Figure 2, we consider large values of r. For even splitting, the fullness regularly hits 50% and trends downward on average as r grows. For deferred even splitting, the fullness is never low, hits 100% regularly, and trends upward. We can conclude that, at least empirically, deferred even splitting is superior to even splitting for large r.

Those are the results of our simulations. What about in theory?

Theoretical results. Our goal is to find a splitting algorithm for every value of r that achieves fill bounded away from 50%. Table 1 and Figure 3 summarize our results. We are able to prove bounds for different parameter regimes and find that different splitting algorithms are better in different ranges of r. Specifically, our best theoretical results for small $r \leq 7B/18$ alternate between two algorithms: even splitting and deferred even splitting. Then, around the value of $B/2$ where these algorithms only achieve 50% fill, we use a form of *uneven* splitting defined in Section 5. Finally, for larger batch sizes $r \geq 2B/3$, we use deferred even splitting.

Remark. We briefly elaborate on the alternation between even splitting and deferred even splitting for small r. For deferred even splitting, for some ranges of r we can use a black box reduction to

[3]In general, δ can be different at each split.

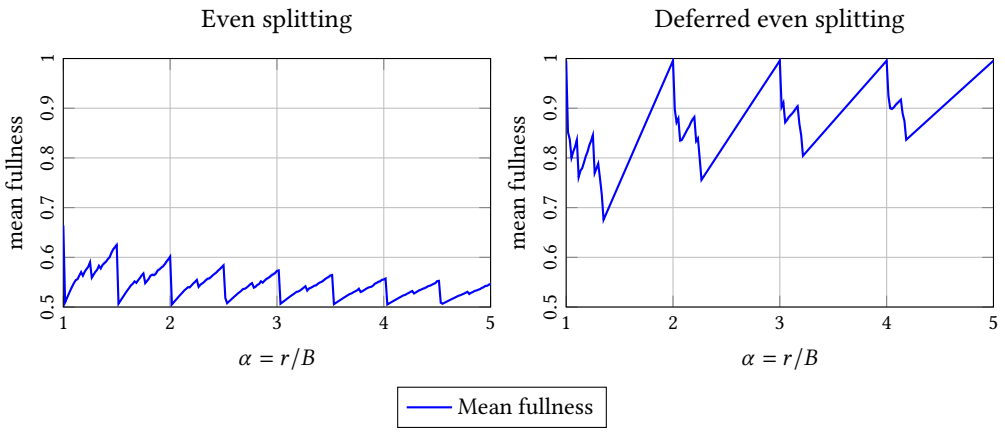


Fig. 2. Experimental fullness of deferred and even splitting on batch insertions on blocks of size $B = 240$. 200,000 insertions are made from empty using batch insertions of varying length ($r \in [B, 5B]$). The mean fullness across 10 independent runs is shown, as a function of $\alpha = r/B$.

r/B	Fill	Fill value range	Splitting Algorithm	Reference
$(1/(2i),\ 1/(2i-1)]$	$\frac{2ir}{B}\ (H_{2i} - H_i)$	$[0.5, 1]$	Deferred even split	Section 6
$[1/B, 0.0058]$	$\ln(2) - \frac{5r}{B}$	$[0.664, 0.693]$	Even split	Section 4.4
$(0.0058, 0.21]$	$\frac{2(B+1)}{3B+1+2r}$	$[0.583, 0.664]$	Even split	Section 4.4
$(0.21, 7/18]$	0.583	0.583	Even split	Section 4.4
$(7/18, 1/2]$	$\frac{3r}{2B}$	$[0.583, 0.750]$	Uneven split	Section 5.1
$(1/2, 2/3]$	$\frac{10r}{9B}$	$[0.555, 0.740]$	Uneven split	Section 5.1
$(2/3, 1]$	$\frac{r}{B}$	$[0.666, 1]$	Deferred even split	Section 5.2
$(1, \infty)$	$\max\left\{\frac{r/B+1/2}{\lceil r/B+1\rceil}, \frac{2}{3}\right\} - \frac{1}{B}$	$[0.666, 1]$	Deferred even split	Section 5.2

Table 1. Fill achieved by our algorithms for different batch sizes

Yao's setting. This occurs when the divisibility properties of the batch size enable us to treat a chunk of insertions as a single insertion in Yao's setting. However, for much of the parameter range, r does not have such nice divisibility properties. In these cases, we analyze even splitting from scratch by extending the analysis of Yao. This analysis constitutes the technical bulk of the paper.

Open problems. Our results leave several open problems:

(1) Can we get tight bounds for even splitting for $r > 1$?
(2) The exact analytical bounds for deferred even splitting on small r have some gaps. Can we extend our approach for analyzing even splitting on small r to achieve bounds for deferred even splitting for values of r to fill the gaps between our bounds (the red lines in Figure 1)?
(3) Is there a splitting algorithm that achieves greater than 50% fill without knowing r?

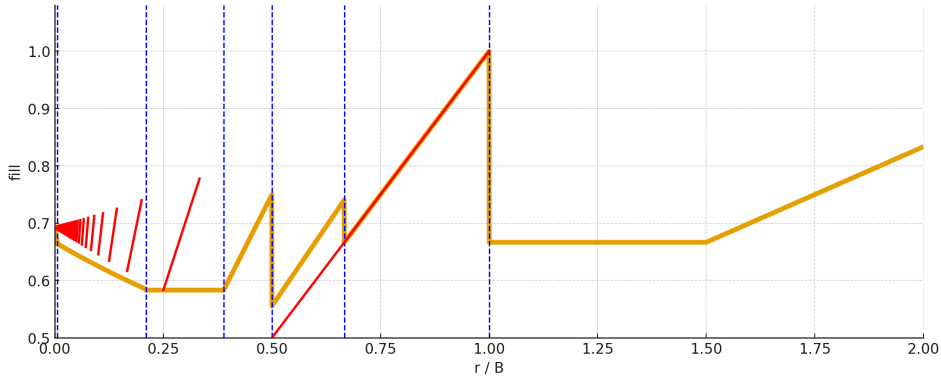


Fig. 3. Our theoretical bounds, presented in Table 1, plotted as a function of r/B. The red segments represents the first line in Table 1, while the continuous orange plot represents the remaining entries.

(4) Is there a splitting algorithm that achieves greater than 50% fill when r can vary between insertions? What about when r is a lower bound on the length of the insertion runs, that is, where the length of the run can vary from insertion run to insertion run, but in each case has at least r items?

Paper organization. To achieve our bounds, the main technical lifting is in achieving bounds for even splitting when r is relatively small (rows 2, 3, and 4 of Table 1). For this reason, Sections 2, 3, and 4 are mostly dedicated to this regime. In Section 2, we formally define the problem and show how to recast it as analyzing the behavior of a certain recurrence. In Section 3, we present a high-level overview of our techniques, including a discussion of the significant challenges involved in extending Yao's analysis for uniformly random insertions to work for batched random insertions. In Section 4, we prove our results for the small-r regime. In Section 5.1, we turn to the medium-r regime and summarize our results for uneven splitting, which use the tools we developed for even splitting but are technically simpler to obtain. In Section 5.2, we summarize our results for the regime of large r, which do not require any machinery. Finally, in Section 6, we present a closed form for the fullness of the deferred even splitting strategy for a range of batch sizes. The details and proofs for the latter three sections were omitted due to space constraints and are presented in the full version of the paper.[4]

2 Preliminaries

As previously discussed, we study the problem of measuring the internal fragmentation of B-trees subject to batches of random insertions under different splitting algorithms. We use the standard formulation of B+ trees [20] in which all keys are stored in the leaves. This affords a simplified model of the problem in which all keys are stored in sorted order across a collection of memory blocks, and the algorithm is specified by two central choices. The first is how to split a memory block. In other words, when an insertion occurs into a block that is full, the algorithm splits it into two memory blocks and chooses how to distribute the keys between the new blocks. The second is how to break ties. Specifically, when an insertion occurs in the interval between the maximum of one memory block and the minimum of the next, the algorithm needs to decide which memory block to insert the key into. Our measure of interest is the average block size in the structure (i.e., total number of keys divided by the number of blocks) as the number of keys in the tree grows.

[4]https://arxiv.org/abs/2603.12211

The rest of this section is primarily dedicated to reviewing the framework we use to analyzing even splitting, as this analysis is the more technically involved part of this work. We formalize and state results for our other splitting procedures, (i.e. uneven and deferred even splitting) in Section 5.

2.1 Uniformly Random Insertions and Batched Random Insertions

In the model studied by Yao [21, 25], at each time step, a key is inserted into a uniformly random position. More precisely, assume the keys inserted thus far into the structure are $k_1, ..., k_n$. These keys induce $n + 1$ disjoint intervals: $(-\infty, k_1), (k_1, k_2), ..., (k_{n-1}, k_n), (k_n, \infty)$.

At each step, the adversary inserts a key into one of these intervals, where each interval has probability $\frac{1}{n+1}$ of receiving the insertion, independently of any prior insertions. As discussed in Section 1, we study a generalization of this workload in which at each step, the adversary does not insert a single key into the interval chosen uniformly at random, but rather r keys provided in ascending order. We proceed to formally describe the even splitting algorithm and to summarize Yao's analysis of even splitting against uniformly random insertions. We then explain how this analysis naturally generalizes to part of our batched random insertion setting.

Even splitting algorithm. In an even splitting algorithm, when an insertion occurs into a full block, i.e. one that contained B keys prior to the insertion, we split the block into two new blocks and disperse the keys evenly. More precisely, we place the smallest $\lfloor (B + 1)/2 \rfloor$ keys into the left block resulting from the split, and the largest $\lceil (B + 1)/2 \rceil$ keys into the right block. For simplicity of presentation, whenever we discuss even splitting, we assume B is odd. For tie breaking, i.e., when an insertion occurs in the interval between the maximum of one memory block and the minimum of the next, we always choose to insert the key into the left block. Note that this choice is arbitrary as the analysis still holds with minor modifications if we make the opposite choice.

2.2 Even Splitting Against Uniformly Random Insertions

We begin by reviewing Yao's analysis for even splitting against uniformly random insertions. To compute the average block size, we must analyze how the composition of blocks changes in our structure. Observe that after the first split occurs, no blocks of size less than $d := \frac{B+1}{2}$ can exist in the data structure for the remainder of time, and thus it suffices to analyze the expected number of blocks of size i for each $i \in \{d, d + 1, ..., 2d - 1 = B\}$.

For all n and $i \in \{d, d + 1, ..., 2d - 1\}$, we let x_i^n denote the number of blocks in the data structure which contain exactly i keys after a total of n keys have been inserted. We refer to this type of block as an 'i-block'. We let v_n denote the vector of the expectation values of each of these random variables, i.e.,

$$v_n = \left(\mathbb{E}\left(x_d^n\right), \mathbb{E}\left(x_{d+1}^n\right), ..., \mathbb{E}\left(x_{2d-1}^n\right)\right).$$

Each component of this vector can be represented by a scalar recurrence.

$$\mathbb{E}\left(x_i^{n+1}\right) = \mathbb{E}\left(x_i^n + \Delta_i^n\right) = \mathbb{E}\left(x_i^n\right) + \mathbb{E}\left(\Delta_i^n\right),$$

where $\Delta_i^n := x_i^n - x_i^{n+1}$ is the difference in the number of blocks of size i from v_n to v_{n+1}.

To completely specify a recurrence, we must characterize Δ_i^n. Observe that if we assume before the sequence of insertions begins that $-\infty$ is inserted into the structure (i.e. a dummy key smaller than all other keys is inserted), then the probability of hitting a particular block of size i when n keys are present in the structure becomes exactly $\frac{i}{n}$. Thus, if a total of n keys have been inserted (including the dummy key), the next insertion occurs in a block of size i with probability exactly $i \cdot \frac{x_i^n}{n}$. Thus, for every $i > d$, we have:

$$\mathbb{E}(\Delta_i^n) = -i \cdot \frac{\mathbb{E}(x_i^n)}{n} + (i - 1) \cdot \frac{\mathbb{E}(x_{i-1}^n)}{n}.$$

This is because if the next insertion occurs in a block of size $i > d$, the block then transforms into a block of size $i+1$ (or is split into two blocks of size d), reducing the number of blocks of size i by 1. Or, if the insertion occurs in a block of size $i-1$, then the number of blocks of size i increases by 1. For the case of $i = d$, if the insertion occurs in a block of size d, then the number of blocks of size $d+1$ is increased by 1. If the insertion occurs in a block of size $2d-1 = B$, then a split is triggered and the number of blocks of size d increases by 2. So we obtain:

$$\mathbb{E}\left(\Delta_d^{n+1}\right) = -d \cdot \frac{\mathbb{E}\left(x_d^n\right)}{n} + 2 \cdot (2d-1) \cdot \frac{\mathbb{E}\left(x_{2d-1}^n\right)}{n}.$$

We thus obtain the following recurrence for v_n: $\forall n \geq d : v_{n+1} = \left(I + \frac{1}{n} \cdot A\right) v_n$, where:

$$A = \begin{bmatrix} -d & 0 & \cdots & \cdots & 0 & 2B \\ d & -(d+1) & \cdots & \cdots & 0 & 0 \\ 0 & d+1 & \cdots & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & -(B-1) & 0 \\ 0 & 0 & \cdots & \cdots & B-1 & -B \end{bmatrix}.$$

2.3 Even Splitting Against Batched Insertions

We now consider the case of batched random insertions. As described above, at each time step, the adversary inserts a batch of $1 \leq r < \frac{B}{2}$ consecutive keys into a randomly chosen interval. We demonstrate how the analysis reviewed in Section 2.2 naturally generalizes to this setting. Like Section 2.2, we consider the vector describing the expected composition of our data structure, or more precisely the expected quantity of each type of block:

$$v_n = \left(\mathbb{E}\left(x_d^n\right), \mathbb{E}\left(x_{d+1}^n\right), ..., \mathbb{E}\left(x_{2d-1}^n\right)\right).$$

We can similarly obtain a scalar recurrence for each component, but it is convenient for analysis to take a step size of r: $\mathbb{E}\left(x_i^{n+r}\right) = \mathbb{E}\left(x_i^n\right) + \mathbb{E}\left(\Delta_i^n\right)$, where now Δ_i^n denotes the expected change in the quantity of blocks of size i between v_n and v_{n+r}. For $i > d$, if the first key in a batch is inserted into a block of size i, then by the end of the batch the number of blocks of size i has decreased by 1, while if instead it hits a block of size $d + (i-d-r) \bmod d$, then by the end of the batch the number of blocks of size i has increased by 1. We can therefore obtain, for every $i > d$, and letting w_i denote $d + (i-d-r) \bmod d$:

$$\mathbb{E}(\Delta_i^n) = -i \cdot \frac{\mathbb{E}(x_i^n)}{n} + w_i \cdot \frac{\mathbb{E}(x_{w_i}^n)}{n}.$$

For the case of $i = d$, if the batch begins in a block of size d, then the number of blocks of size d is decreased by 1. If the batch begins in a block of size $B+1-r$, then the last key in the batch triggers a split, and thus the number of blocks of size d is increased by 2. If the batch begins in a block of size $k > B+1-r$, then a split occurs during the batch, and following the split additional keys are inserted into one of the created siblings. Thus, the number of blocks of size d is increased by 1. We therefore obtain:

$$\mathbb{E}(\Delta_d^n) = -d \cdot \frac{\mathbb{E}(x_d^n)}{n} + 2 \cdot (B+1-r) \cdot \frac{\mathbb{E}(x_{B+1-r}^n)}{n} + \sum_{B+1-r<k\leq B} k \cdot \frac{\mathbb{E}(x_k^n)}{n}. \tag{1}$$

We therefore obtain a matrix recurrence similar to the one in Section 2.2: $\forall n \geq n_0 : v_{n+r} = \left(I + \frac{1}{n} \cdot A(B,r)\right) v_n$, where n_0 is the minimum multiple of r that is at least d. We use $A(B,r)$ to denote the ***transition matrix*** for blocks of capacity B and batches of size r (observe that we've specified

$A(B, 1)$ in Section 2.2). We now turn to fully specify $A(B, r)$ based on the scalar recurrences above. Before we do so, we present a non-standard indexing method that will be convenient for later analyses.

2.3.1 Indexing method. As discussed, following the first split, all blocks in our structure have size at least d, which is why it suffices to track blocks of at least this size. This is reflected in the fact that the first entry in v_n corresponds to the number of blocks of size d and the last entry corresponds to the number of blocks of size B. Similarly, the first row in $A(B, 1)$ (specified in Section 2.2) corresponds to the recurrence relation for the expected number of blocks of size d. It is convenient for the purpose of later analysis to use indexing that reflects this - i.e to index entries in d-dimensional vectors, and $d \times d$-dimensional matrices by the block sizes they correspond to.

To describe this formally, we first define $f : \{0, ..., d-1\} \to \{d, ..., 2d-1\}$ by $f(i) = d + i$. Note that f is a bijective mapping. Let B be any $d \times d$ matrix, and w be a d-dimensional vector. We will use $f(i)$ to index the ith row/column in B and the ith component of w. For example, $B_{i,j}$ refers to $B_{i-d,j-d}$ in standard indexing. For a principle submatrix P of a $d \times d$ matrix B, we will use $f(i)$ to index the j^{th} row/column in P, where i is the corresponding index to j in the greater matrix B. Similarly, for u, a subvector of a d-dimensional vector w, we will use $f(i)$ to index the j^{th} entry, where i is the corresponding index to j in w.

2.3.2 Specifying the general transition matrix. We now specify the general transition matrix $A(B, r)$ using the scalar recurrences presented in Section 2.3 and our non-standard indexing method. The first row is described separately as it has a unique structure. Furthermore, it is convenient to split the remaining rows into two categories so that we can explicitly specify the entries without using the modulo operation. All entries not explicitly mentioned below are zero. Below, A always denotes $A(B, r)$; recall that we are assuming that $r < B/2$.

First row:

$$\begin{aligned} &A_{d,d} = -d, \\ &A_{d,B+1-r} = 2 \cdot (B + 1 - r), \\ &\text{for every } B + 1 - r < k \leq B : A_{d,k} = k. \end{aligned}$$

Wraparound rows:

$$\begin{aligned} &\forall d < k \leq d + r - 1 : \\ &\quad A_{k,k} = -k, \\ &\quad A_{k,d+k-r} = d + k - r \end{aligned}$$

Remaining rows:

$$\begin{aligned} &\forall d + r \leq k \leq B : \\ &\qquad A_{k,k} = -k, \\ &\qquad A_{k,k-r} = k - r. \end{aligned}$$

Observe that $A_{i,j}$ can be described as follows: $\frac{A_{i,j}}{j}$ is the signed difference that occurs in the number of blocks of size i if a batch of insertions begins in a block of size j.

For a concrete example that shows how we obtain this matrix directly from the recurrence relations we specified, we demonstrate how the first row in A is obtained. As defined in Section 2.3:

$$\mathbb{E}\left(x_d^{n+r}\right) = \mathbb{E}\left(x_d^n\right) + \mathbb{E}\left(\Delta_i^n\right).$$

Therefore, if we denote the first row of A by $\vec{a}$, for our desired recurrence to be fulfilled, it needs to satisfy (letting $\langle , \rangle$ denote the standard inner product):

$$\langle e_1 + \frac{1}{n} \cdot \vec{a}, v_n \rangle = \mathbb{E}\left(x_d^n\right) + \mathbb{E}\left(\Delta_d^n\right) \iff \langle \vec{a} \cdot v_n \rangle = n \cdot \mathbb{E}\left(\Delta_d^n\right). \tag{2}$$

Now, by Equation (1):

$$n \cdot \mathbb{E}\left(\Delta_d^n\right) = \mathbb{E}(\Delta_d^n) = -d \cdot \mathbb{E}(x_d^n) + 2 \cdot (B + 1 - r) \cdot \mathbb{E}(x_{B+1-r}^n) + \sum_{B+1-r<k\leq B} k \cdot \mathbb{E}(x_k^n) = \langle u, v_n \rangle,$$

where $u = (-d, 0, ..., 0, 2(B + 1 - r), B + 2 - r, ...B)$. So, the first row of A, $\vec{a}$, needs to satisfy:

$$\langle \vec{a}, v_n \rangle = \langle u, v_n \rangle.$$

So, setting $\vec{a} = u$ satisfies Equation (2).

3 Technical Overview

We begin by providing a technical overview of our contributions. We spend the majority of this section giving a technical overview of the analysis for even splitting in B-trees against uniformly random batched insertions, as it is the most technically challenging. As outlined in Section 2, the recurrence describing the behavior of even splitting subject to batched random insertions of size r is of the form $v_{n+r} = \left(I + \frac{1}{n} \cdot A(B, r)\right) v_n$, where $A(B, r)$ is the transition matrix. For $r = 1$ we get the exact recurrence analyzed by Yao [25].

Main objective. Our goal is to compute the asymptotic average block size in the data structure. Observe that in fact v_n/n can be used to characterize the average block size (see Section 4.3 for more details). This is intuitive, since the ratio between different entries in v_n/n tells us the ratio between the quantities of blocks of different sizes in our structure. So, our goal from now on is to analyze $\lim_{n\to\infty} v_n/n$. It is not hard to verify that if v_n/n admits a steady state (i.e. $v_{n+r}/(n + r) = v_n/n$ for sufficiently large n) then that steady state must be an eigenvector of $A(B, r)$ corresponding to eigenvalue r. Thus it is natural to expect that v_n/n should converge to an eigenvector of $A(B, r)$. The key technical challenges in our work have to do with proving this intuition holds, and as we'll see, analyzing the structure of an eigenvector of $A(B, r)$.

Yao's approach. The analysis of random B-trees provided in [25] can be interpreted as consisting of two primary stages:

(1) The first step involves proving that $\frac{v_n}{n}$ converges to an eigenvector of $A(B, 1) \coloneqq A$. As discussed above, this is useful as $\frac{v_n}{n}$ can be used to represent the expected average block size.
(2) Then, the eigenvector is computed exactly and is used to obtain the average block size.

Challenges in generalizing Yao's approach. Much like Yao, our general approach is to prove that each recurrence $v_{n+r} = \left(I + \frac{1}{n} \cdot A(B, r)\right) v_n$ converges to an eigenvector of $A(B, r)$, and then to analyze this eigenvector. However, Yao's methods do not easily apply to our setting, and we thus deviate significantly in how we go about these tasks. The difficulties in generalizing from Yao's analysis are as follows:

First Challenge: Proving that v_n/n converges to an eigenvector of $A(B, r)$ for every r is a significantly harder task when $r > 1$. Yao's approach for $A = A(B, 1)$ relied on a low-level analysis that explicitly computed specific information about this particular matrix A; for example, his analysis required computing the characteristic polynomial of A and factoring it according to a specific form. But the moment we turn to $r > 1$, such an approach is quite difficult to execute for two reasons.

First, even if we could repeat Yao's factorization of the characteristic polynomial for a specific value of r, it is unclear that there should exist a general formula for the characteristic polynomial that can be parameterized by r. As suggested by Figure 1, the characteristics of $A(B, r)$ seem to vary in a non-monotonic way with r, so there is no reason to assume a single formulation can work for all values of r simultaneously.

Second, even if we could achieve a general formulation for the characteristic polynomial and obtain a complete spectral characterization of the transition matrix for general r, arguing about convergence for general $r > 1$ requires different techniques from $r = 1$. Specifically, $A(B, 1)$ is a diagonalizable matrix, which affords decomposing the initial state into a basis of eigenvectors and computing limits on each component separately. $A(B, r)$, on the other hand, is generally not diagonalizable; for example $A(15, 4)$ is not a diagonalizable matrix.

Second Challenge: Even if we know the (normalized) recurrence converges to an eigenvector, analyzing this eigenvector in order to derive bounds is much more involved when $r > 1$. For $r = 1$, the eigenvector admits a neat closed form and thus the asymptotic average block size can be computed precisely. By contrast, for $r > 1$ we encounter two similar difficulties to the previous step. First, considering the non-smooth nature of the curve in Figure 1 and the fact that the eigenvector can be used to express the average block size (see Section 4.3 for details), we should not expect a closed form that holds for every value of r simultaneously. Second, even if we set a specific $r > 1$ and try to characterize the eigenvector for this particular $A(B, r)$, the system of equations for the eigenvector that is generated by $A(B, r)$ is significantly more complicated than that of $A(B, 1)$ due to its different structure, and there is thus no reason to assume it admits a neat closed form (as far as we can tell it does not).

Overcoming the challenges for generalization. We now discuss how we overcome each of the above challenges.

First Challenge: To prove v_n/n converges, we first avoid any precise analysis of the characteristic polynomial and instead rely on simple structural characteristics that hold for all $A(B, r)$ in order to characterize their spectrum. Then, we use this characterization to argue about convergence through the complex Jordan form of $A(B, r)$. See Section 4.2 for details.

Second Challenge: We avoid the need to compute the eigenvector precisely and provide looser characterizations in order to derive lower bounds on the average block size . We observe that while the eigenvector does not admit a neat closed form, certain subvectors of the eigenvector are much more structured (although the way in which the values of these subvectors relate to each other is unclear). We then analyze these subvectors in order to provide lower bounds for the average block size. See Section 4.4 for details.

3.1 Algorithms for Larger r

The discussion thus far has pertained to even splitting, which produces favorable bounds for $r < 7B/18$ (see Table 1 for more details). As discussed in Section 1, our bounds for even splitting decay as r approaches $B/2$, and we thus appeal to a different set of algorithms for larger values of r. These algorithms are less technically involved to analyze; we now provide a brief overview.

Moderately large $\boldsymbol{r}$: $\frac{7}{18}\boldsymbol{B} \leq \boldsymbol{r} \leq \frac{2}{3}\boldsymbol{B}$. In this regime, we obtain favorable bounds by using uneven splitting algorithms. These algorithms are easier to analyze because by choosing specific appropriate splits, we are able to confine the possible sizes of blocks in our structure at the end of each insertion batch to a small predetermined set. In particular, we are able to ensure that there are at most three

possible blocks sizes; for example, for $r = B/2$, our algorithms will ensure that all blocks have size $B/4, B/2$ or $3B/4$. We can then readily analyze the recurrence using tools we developed for the analysis of even splitting. These tools are simple to apply because our transition matrix has size at most 3×3, as we have at most 3 different possible block sizes. See Section 5 for more details.

Large values of $\boldsymbol{r}$*:* $\boldsymbol{r > \frac{2}{3}B}$*.* This is by far the easiest case; no linear algebra is required to conclude that deferred even splitting ensures good space utilization. See Section 5 for details.

4 Analysis of Even Splitting Against Batched Insertions

Our goal in this section is to establish the bounds presented in Table 1 for even splitting. The organization of the analysis is as follows: Section 4.1 provides a spectral characterization of our transition matrices $A(B, r)$ (see Section 2 for a specification of these matrices). Then, in Section 4.2, we use this characterization to prove the normalized recurrence relations presented in Section 2 converge to an eigenvector of the transition matrix. Section 4.3 demonstrates how any eigenvector of the transition matrix can be used to compute the limit of the average block size. Then, we spend Section 4.4 analyzing the structure of an eigenvector to derive lower bounds for the asymptotic average block size.

4.1 Spectral Characterization of the Transition Matrix

4.1.1 Spectral Properties of Irreducible Metzler Matrices. Our analysis uses a well known characterization of the spectrum of irreducible Metzler matrices. For completeness, we begin with a self-contained review of the definition of Metzler matrices and a standard proof. For further reading, refer to [14].

Definition 4.1. A square matrix $A \in \mathbb{R}^{d \times d}$ is a Metzler matrix if and only if all off-diagonal entries in A are nonnegative.

Definition 4.2. For a square matrix $A \in \mathbb{R}^{d \times d}$, we define the underlying directed graph of A, denoted by $G[A]$, as a directed graph on $V = \{0, ..., d-1\}$ in which for every $i, j \in \{0, ..., d-1\}$, there exists a directed edge from i to j in $G[A]$ if and only if $A_{i,j} \neq 0$.

Definition 4.3. We say that a square matrix A is irreducible if $G[A]$ is strongly connected.

Lemma 4.4. *Let $A \in \mathbb{R}^{d \times d}$ be an irreducible Metzler matrix, and assume there exists an entrywise positive vector $w \in \mathbb{R}^d$ such that $w^T \cdot A = r \cdot w^T$ for some $r > 0$. Then:*

(1) r is a simple eigenvalue of A, and there exists a positive eigenvector corresponding to it.
(2) For every other eigenvalue $\lambda \in \sigma(A)$, we have $\mathcal{R}e(\lambda) < r$, where $\mathcal{R}e(\cdot)$ denotes the real part of a complex number.

Proof. Let $A \in \mathbb{R}^{d \times d}$ be an irreducible Metzler matrix, and assume $w^T \cdot A = r \cdot w^T$ for a positive w. Note that this already means r is an eigenvalue of A. Now, let $B = A + c \cdot I$ where

$$c = max\left\{\left|A_{i,i}\right| : i \in \{0, ..., d-1\}\right\}.$$

Observe that B is a nonnegative, irreducible matrix. Further:

$$w^T \cdot B = w^T \cdot (A + c \cdot I) = r \cdot w^T + c \cdot w^T = (r + c) \cdot w^T.$$

So, w^T is a positive left eigenvector of B with eigenvalue $r' = r + c > 0$. By applying Perron-Frobenius theorems [14] for nonnegative irreducible matrices, we conclude the following:

- r' is the spectral radius of B. This implies that for any other eigenvalue λ of B, we have $\mathcal{R}e(\lambda) < r'$.
- r' is a simple eigenvalue, and there exists a corresponding positive eigenvector of B.

Now, since the eigenspace of A corresponding to r is equal to the eigenspace of B corresponding to r', we can conclude that r is simple, and that there exists a positive eigenvector of A corresponding to it. Furthermore, let $\lambda \in \sigma(A)$. Then $\lambda + c \in \sigma(B)$, and so

$$\begin{aligned} &\mathcal{R}e(\lambda + c) < r' \\ \implies &\mathcal{R}e(\lambda) + c < r + c \\ \implies &\mathcal{R}e(\lambda) < r, \end{aligned}$$

as needed. □

4.1.2 Analyzing the Spectral Properties of $A(B, r)$. For every $k \in \{d, ..., 2d-1\}$, let $\Delta_k \in \{-1, 0, 1, 2\}^d$ be the vector describing the change in v_n when a batch of insertions hits a block of size k. In other words, $\Delta_k = v_{n+r} - v_n$ (with coordinate-wise subtraction) provided that the next batch occurs in a block of size k. We prove next that for each k, the kth column of the transition matrix is equal to $\Delta_k \cdot k$. To see why this is intuitive, observe that the kth column has values in $\{-1, 0, 1, 2\}$ when normalized by k. We have $A_{k,k} = -1$, which corresponds to decreasing the number of blocks of size k when we hit one. Furthermore, for all the remaining entries, $A_{i,k} \neq 0$ if and only if hitting a block of size k increases the number of blocks of size i by 1 or 2. This is due to our basic model; the ith row reflects how the number of blocks of size i changes. We proceed to state and prove this formally.

Lemma 4.5. *The kth column of $A(B, r)$ is equal to $\Delta_k \cdot k$.*

Proof. For $d \leq k < 2d - r$, we have $A_{k,k} = -k$ and $a_{k+r,k} = k$, and indeed Δ_k describes a reduction in the number of blocks of size k and an increase in the number of blocks of size $k + r$. For $k = 2d - r$ we have $A_{k,k} = -k$, and $A_{d,k} = 2 \cdot k$, and indeed Δ_k describes a reduction in the number of blocks of size k and an increase by 2 in the number of blocks of size d. For $2d - r < k \leq 2d - 1$ we have $A_{k,k} = -k$, $A_{d,k} = k$, and $A_{k+r-(B+1)+d,k} = k$, and indeed Δ_k describes a reduction in the number of blocks of size k and an increase by 1 in both the number of blocks of size d, and the number of blocks of size $k + r - (B + 1) + d$. □

Lemma 4.6. *The value r is an eigenvalue of the transition matrix $A(B, r)$, and the matrix has an entrywise positive left eigenvector corresponding to r.*

Proof. We demonstrate a positive left eigenvector corresponding to eigenvalue r: let $w = (d, ..., 2d - 1)$, and note that by our indexing method $w_k = k$. By Lemma 4.5,

$$\left[w^T \cdot A(B, r)\right]_k = w^T \cdot k \cdot \Delta_k = k \cdot \left(w^T \cdot \Delta_k\right),$$

where $[\cdot]_k$ is used to denote the kth column of a matrix. Observe that $w^T \cdot \Delta k$ is exactly equal to the net change in the number of elements when inserting r elements if a block of size k is hit, which is invariably equal to r. So overall, we have $\left[w^T \cdot A(B, r)\right]_k = k \cdot r = r \cdot \left[w^T\right]_k$. □

Definition 4.7. Let $\langle r \rangle$ be the cyclic subgroup of $\mathbb{Z}_d$ generated by r, and we let S denote $d + \langle r \rangle \coloneqq \{d + x : x \in \langle r \rangle\}$.

Lemma 4.8. *S is a cycle in $G[A(B, r)]$.*

Proof. It suffices to that demonstrate S is a cycle in $G\left[A(B, r)^T\right]$. To that end, we show that for each $d + x \in S$, there exists an edge from $d + x$ to $d + (x + r) \bmod d$ in $G\left[A(B, r)^T\right]$.

Let $k = d + x \in S$. We know that the kth column in $A(B, r)$ is Δ^k, i.e., the change we see when a batch of insertions hits a block of size k. Hitting a block of size k increases the number of blocks of size $d + (x + r) \bmod d \in S$, and thus the $(d + (x + r) \bmod d)$-th component of the kth column

is nonzero, and therefore we have an edge from $d + x$ to $d + (x + r) \bmod d$ in $G\left[A(B,r)^T\right]$, as needed. □

Corollary 4.9. *Let $A(B,r)_S$ be the principal submatrix of $A(B,r)$ corresponding to S. Then $G[A(B,r)_S]$ is strongly connected.*

The corollary follows from the fact that $G[A(B,r)_S]$ is the induced subgraph of $G[A(B,r)]$ on the vertex set S, and the fact that $G[A(B,r)]$ has a cycle on vertex set S. Therefore, $G[A(B,r)_S]$ is strongly connected (however, in general it is not simply a cycle graph).

Lemma 4.10. *We have $r \in \sigma(A(B,r)_S)$, and furthermore, $A(B,r)_S$ has an entrywise positive left eigenvector corresponding to r.*

Proof. We know that Δ^k reflects the change in blocks we see when an insertion hits a block of size k. Observe that when a block of size $k \in S$ is hit, we only see changes in the quantities of blocks whose sizes are in S. More specifically, if $k = d + x \in S$, the only nonzero entries are k, $d + (x + r) \bmod d \in S$, and possibly $d \in S$. Overall, in this case Δ^k only has support on S.

Therefore, for every $k \in S$, and for $w = (d, ..., 2d - 1)$, we have:

$$(w_S)^T [A(B,r)_S]_k = w^T [A(B,r)]_k .$$

Now, according to the proof of Lemma 4.6 w is a left eigenvector of $A(B,r)$, thus:

$$(w_S)^T [A(B,r)_S]_k = r \cdot \left(w^T\right)_k = r \cdot \left((w_S)^T\right)_k .$$

Overall, $(w_S)^T$ is a positive left eigenvector of $A(B,r)_S$ with eigenvalue r, as needed. □

Lemma 4.11. *The eigenvalue r is a simple eigenvalue of $A(B,r)_S$, for which there exists a positive (right) eigenvector, and for every other eigenvalue λ we have $\mathcal{R}e(\lambda) < r$.*

Proof. As a principal submatrix of a Metzler matrix, $A(B,r)_S$ is a Metzler matrix. According to Corollary 4.9, $G[A(B,r)_S]$ is strongly connected, and therefore $A(B,r)_S$ is an irreducible Metzler matrix. We have demonstrated an entrywise positive w such that $w^T \cdot (A(B,r)_S) = r \cdot w^T$. Therefore by Lemma 4.4, we can conclude that r is simple, has a positive eigenvector, and for every other eigenvalue λ we have $\mathcal{R}e(\lambda) < r$. □

4.2 Proving our recurrences converge

As presented in Section 2, we model the behavior of even splitting through recurrences of the form $v_{n+r} = \left(I + \frac{1}{n} \cdot A(B,r)\right) \cdot v_n$. Having provided spectral characterizations for $A(B,r)$ in Section 4.1, we spend this section proving that under these characterizations our recurrence relations converge (with some normalization).

Lemma 4.12. *Let n_0, r be positive integers, and let $\lambda \in \mathbb{C}$. Then:*

$$\left| \prod_{k=\lceil |\lambda| \rceil}^{m-1} \left(1 + \frac{\lambda}{n_0 + k \cdot r}\right) \right| \leq m^{\frac{\mathcal{R}e(\lambda)}{r}} \cdot \alpha_{\lambda,n_0,r},$$

where $\alpha_{\lambda,n_0,r}$ depends only on $\lambda, n_0,$ and r.

Proof. Observe that for any complex t we have:

$$|1 + t| = \sqrt{1 + 2\mathcal{R}e(t) + |t|^2} \leq \sqrt{\exp\left(2\mathcal{R}e(t) + |t|^2\right)} \leq \exp\left(\mathcal{R}e(t) + \frac{1}{2} \cdot \left|t^2\right|\right).$$

where $\exp(x) \coloneqq e^x$ and $\mathcal{R}e(\lambda)$ denotes the real part of λ.

Therefore, for $t = \frac{\lambda}{n_0+k\cdot r}$ we have:

$$\ln\left(\left|\left(1+\frac{\lambda}{n_0+k\cdot r}\right)\right|\right) \le \ln\left(\exp\left(\mathcal{R}e\left(t\right)+\frac{1}{2}\cdot\left|t^2\right|\right)\right). \tag{3}$$

Now observe:

$$\left|\prod_{k=\lceil|\lambda|\rceil}^{m-1}\left(1+\frac{\lambda}{n_0+k\cdot r}\right)\right| = \prod_{k=\lceil|\lambda|\rceil}^{m-1}\left|\left(1+\frac{\lambda}{n_0+k\cdot r}\right)\right| = \exp\left(\sum_{k=\lceil|\lambda|\rceil}^{m-1}\ln\left(\left|\left(1+\frac{\lambda}{n_0+k\cdot r}\right)\right|\right)\right).$$

Therefore, according to Equation (3), we get:

$$\left|\prod_{k=\lceil|\lambda|\rceil}^{m-1}\left(1+\frac{\lambda}{n_0+k\cdot r}\right)\right| \le \exp\left(\sum_{k=\lceil|\lambda|\rceil}^{m-1}\mathcal{R}e\left(t\right)+\frac{1}{2}\cdot\sum_{k=1}^{m-1}\left|t^2\right|\right).$$

We bound both sums in the exponent separately, beginning with $\sum_{k=\lceil|\lambda|\rceil}^{m-1}\mathcal{R}e\left(t\right)$:

$$\sum_{k=\lceil|\lambda|\rceil}^{m-1}\mathcal{R}e\left(t\right) = \mathcal{R}e\left(\lambda\right)\cdot\sum_{k=\lceil|\lambda|\rceil}^{m-1}\frac{1}{n_0+k\cdot r} \coloneqq \mathcal{R}e\left(\lambda\right)\cdot\varphi\left(m\right),$$

and

$$\int_{\lceil|\lambda|\rceil}^{m}\frac{dx}{n_0+x\cdot r} \le \varphi\left(m\right) \le \frac{1}{n_0+L\cdot r}+\int_{\lceil|\lambda|\rceil}^{m-1}\frac{dx}{n_0+x\cdot r}.$$

To evaluate the lower bound, we have:

$$\int_{\lceil|\lambda|\rceil}^{m}\frac{dx}{n_0+x\cdot r} = \frac{1}{r}\cdot\ln\left(\frac{n_0+m\cdot r}{n_0+\lceil|\lambda|\rceil\cdot r}\right) \ge \frac{1}{r}\cdot\ln\left(m\right)+\gamma_{n_0,\lambda,r},$$

where $\gamma_{n_0,\lambda,r}$ depends only on n_0, λ, and r. To evaluate the upper bound, we have:

$$\frac{1}{n_0+L\cdot r}+\int_{\lceil|\lambda|\rceil}^{m-1}\frac{dx}{n_0+x\cdot r} = \frac{1}{n_0+\lceil|\lambda|\rceil\cdot r}+\frac{1}{r}\cdot\ln\left(\frac{n_0+(m-1)\cdot r}{n_0+\lceil|\lambda|\rceil\cdot r}\right) \le \frac{1}{r}\cdot\ln\left(m\right)+\gamma'_{n_0,\lambda,r},$$

where $\gamma'_{n_0,\lambda,r}$ depends only on n_0, λ, and r. This implies:

$$\frac{1}{r}\cdot\ln\left(m\right)+\gamma_{n_0,\lambda,r} \le \varphi\left(m\right) \le \frac{1}{r}\cdot\ln\left(m\right)+\gamma'_{n_0,\lambda,r}.$$

So overall, we have:

$$\sum_{k=\lceil|\lambda|\rceil}^{m-1}\mathcal{R}e\left(t\right) = \mathcal{R}e\left(\lambda\right)\cdot\varphi\left(m\right) \le \mathcal{R}e\left(\lambda\right)\left(\frac{1}{r}\cdot\ln\left(m\right)+\beta_{n_0,\lambda,r}\right), \tag{4}$$

where

$$\beta_{n_0,\lambda,r} = \begin{cases}\gamma'_{n_0,\lambda,r} & \mathcal{R}e\left(\lambda\right)\ge 0\\ \gamma_{n_0,\lambda,r} & \mathcal{R}e\left(\lambda\right)<0.\end{cases}$$

Note that $\beta_{n_0,\lambda,r}$ only depends on n_0, λ, and r and is in particular independent of m. Now to bound $\frac{1}{2}\cdot\sum_{k=1}^{m-1}\left|t^2\right|$, we have:

$$\frac{1}{2}\cdot\sum_{k=\lceil|\lambda|\rceil}^{m-1}\left|t^2\right| = \frac{1}{2}\cdot|\lambda|^2\cdot\sum_{k=\lceil|\lambda|\rceil}^{m-1}\frac{1}{\left(n_0+k\cdot r\right)^2} \le \frac{1}{2}\cdot|\lambda|^2\cdot\sum_{i=1}^{\infty}\frac{1}{i^2} \le |\lambda|^2. \tag{5}$$

Overall, applying Equation (4) and Equation (5) we achieve:

$$\exp\left(\sum_{k=\lceil|\lambda|\rceil}^{m-1}\mathcal{R}e\,(t)+\frac{1}{2}\cdot\sum_{k=\lceil|\lambda|\rceil}^{m-1}\left|t^2\right|\right)\leq\exp\left(\frac{\mathcal{R}e\,(\lambda)}{r}\cdot\ln(m)+\mathcal{R}e\,(\lambda)\cdot\beta_{n_0,\lambda,r}+|\lambda|^2\right)$$
$$=m^{\frac{\mathcal{R}e(\lambda)}{r}}\cdot\alpha_{n_0,\lambda,r}.$$

□

Lemma 4.12 tells us that, up to a constant (with respect to m), the contribution of eigenvalue λ behaves like a power of m with exponent $\mathcal{R}e(\lambda)/r$. For full Jordan blocks, we need a companion bound for the nilpotent part, which turns out to contribute only polylogarithmic growth.

Lemma 4.13. *Let n_0, r be positive integers, let $\lambda\in\mathbb{C}$, and let N_λ be a nilpotent matrix ($N_\lambda^d=0$). Then:*

$$\left|\prod_{k=\lceil|\lambda|\rceil}^{m-1}\left(I+\frac{N_\lambda}{n_0+k\cdot r+\lambda}\right)\right|\leq C_{N_\lambda}\cdot(\ln(m))^d$$

for sufficiently large m, where C_{N_λ} depends only on N_λ and is independent of m (where $|\cdot|$ refers to the standard operator norm for complex matrices).

Proof. First if we let $a_k=\frac{1}{n_0+k\cdot r+\lambda}$, then:

$$\left|\prod_{k=\lceil|\lambda|\rceil}^{m-1}\left(I+\frac{N_\lambda}{n_0+k\cdot r+\lambda}\right)\right|=\left|\prod_{k=\lceil|\lambda|\rceil}^{m-1}(I+a_k\cdot N_\lambda)\right|.$$

Observe that all matrices in the product commute and thus:

$$\left|\prod_{k=\lceil|\lambda|\rceil}^{m-1}(I+a_k\cdot N_\lambda)\right|\leq\left|I+\sum_{j=1}^{d-1}p_j(\vec{a})\cdot(N_\lambda)^j\right|,\tag{6}$$

where

$$\forall j\geq 1: p_j(\vec{a})=\sum_{\lceil|\lambda|\rceil\leq k_1<\ldots<k_j\leq m-1}a_{k_1}\cdot\ldots\cdot a_{k_j}.$$

It would thus suffice to provide the desired upper bound for the right hand side of Equation (6), which we will focus on for the remainder of the proof. By the triangle inequality and submultiplicativity of the operator norm:

$$\left|I+\sum_{j=1}^{d-1}p_j(\vec{a})\cdot(N_\lambda)^j\right|\leq 1+\sum_{j=1}^{d-1}\left|p_j(\vec{a})\right|\cdot|N_\lambda|^j.\tag{7}$$

Using a standard counting argument:

$$\left|p_j(\vec{a})\right|=\left|\sum_{\lceil|\lambda|\rceil\leq k_1<\ldots<k_j\leq m-1}a_{k_1}\cdot\ldots\cdot a_{k_j}\right|\leq_\triangle\sum_{\lceil|\lambda|\rceil\leq k_1<\ldots<k_j\leq m-1}\left|a_{k_1}\right|\cdot\ldots\cdot\left|a_{k_j}\right|$$
$$\leq\frac{1}{j!}\cdot\left(\sum_{k=\lceil|\lambda|\rceil}^{m-1}|a_k|\right)^j.$$

Therefore,

$$1 + \sum_{j=1}^{d-1} \left| p_j\left(\vec{a}\right) \right| \cdot |N_\lambda|^j \leq 1 + \sum_{j=1}^{d-1} \frac{1}{j!} \cdot \left(\sum_{k=\lceil |\lambda| \rceil}^{m-1} |a_k| \right)^j \cdot |N_\lambda|^j . \tag{8}$$

Now to bound $\sum_{k=\lceil |\lambda| \rceil}^{m-1} |a_k|$, we have:

$$\begin{aligned} \sum_{k=\lceil |\lambda| \rceil}^{m-1} |a_k| &= \sum_{k=\lceil |\lambda| \rceil}^{m-1} \frac{1}{|n_0 + k \cdot r + \lambda|} \leq \sum_{k=\lceil |\lambda| \rceil}^{m-1} \frac{1}{n_0 + k \cdot r - \lceil |\lambda| \rceil} \\ &\leq \sum_{i=1}^{n_0 + m \cdot r} \frac{1}{i} \leq \ln\left(n_0 + m \cdot r\right) + 2 \leq 2 \cdot \ln\left(m\right) . \end{aligned}$$

Therefore:

$$1 + \sum_{j=1}^{d-1} \frac{1}{j!} \cdot \left(\sum_{k=\lceil |\lambda| \rceil}^{m-1} |a_k| \right)^j \cdot |N_\lambda|^j \leq 1 + \sum_{j=1}^{d-1} \frac{1}{j!} \cdot \left(2 \ln\left(m\right)\right)^j \cdot |N_\lambda|^j .$$

This is at most $\left(\ln\left(m\right)\right)^d \cdot C_{N_\lambda}$, where C_{N_λ} depends only on N_λ (specifically on its operator norm and d). Thus, chaining the inequalities from Equation (7) and Equation (8) completes the proof. □

Our convergence theorem uses the notion of a spectral projection. In short, a spectral projection is a projection onto a generalized eigenspace of a matrix. In our recurrences, as we've demonstrated in Section 4.1, r is a simple eigenvalue of the matrix. And thus, the generalized eigenspace corresponding to r is just the eigenspace corresponding to r. We use the fact that the spectral projection for eigenvalue λ can be represented by taking the Jordan form of the matrix: SJS^{-1}, replacing all Jordan blocks corresponding to λ by the identity, and setting all other blocks to the zero matrix. For further reading, refer to [14].

Theorem 4.14. *Assume a matrix recurrence:* $\forall n \geq n_0 : v_{n+r} = \left(I + \frac{1}{n} \cdot A\right) v_n$ *where* $r \in \sigma\left(A\right)$ *is a simple positive eigenvalue, and all other eigenvalues* λ *satisfy* $\mathcal{R}e\left(\lambda\right) < r$. *Then* $\lim_{m \to \infty} \frac{v_{n_0 + m \cdot r}}{n_0 + m \cdot r} = P_r^A \left(\frac{v_{n_0}}{n_0} \right)$, *where* P_r^A *is the spectral projection operator onto* A*'s generalized eigenspace corresponding to eigenvalue* r.

Proof. Iterating the recurrence, we get

$$\begin{aligned} v_{n_0 + m \cdot r} &= \left(\prod_{k=0}^{m-1} \left(I + \frac{1}{n_0 + k \cdot r} \cdot A \right) \right) \cdot v_{n_0} \\ \implies \frac{v_{n_0 + m \cdot r}}{n_0 + m \cdot r} &= \left(\frac{1}{n_0 + m \cdot r} \cdot \prod_{k=0}^{m-1} \left(I + \frac{1}{n_0 + k \cdot r} \cdot A \right) \right) \cdot v_{n_0} . \end{aligned}$$

□

We turn our focus to analyzing the matrix product $P_m = \frac{1}{n_0 + m \cdot r} \cdot \prod_{k=0}^{m-1} \left(I + \frac{1}{n_0 + k \cdot r} \cdot A \right)$.

Write $\sigma\left(A\right) = \{\lambda_1 = r, \lambda_2, ..., \lambda_\ell\}$ and let $A = S \cdot J \cdot S^{-1}$ be the complex Jordan form for A, where J is a block diagonal matrix with blocks $\left(J_{\lambda_1}, ..., J_{\lambda_\ell}\right)$ and $J_{\lambda_i} = \lambda_i \cdot I + N_i$ is the block corresponding

to eigenvalue λ_i. Then

$$P_m = \frac{1}{n_0 + m \cdot r} \cdot \prod_{k=0}^{m-1} \left(I + \frac{1}{n_0 + k \cdot r} \cdot S \cdot J \cdot S^{-1} \right) = \frac{1}{n_0 + m \cdot r} \cdot \prod_{k=0}^{m-1} \left(S \cdot \left(I + \frac{1}{n_0 + k \cdot r} \cdot J \right) \cdot S^{-1} \right)$$
$$= S \cdot \left(\frac{1}{n_0 + m \cdot r} \cdot \prod_{k=0}^{m-1} \left(I + \frac{1}{n_0 + k \cdot r} \cdot J \right) \right) \cdot S^{-1}.$$

Now, since J is a block-diagonal matrix, so is $I + \frac{1}{n_0+k\cdot r} \cdot J$ for each $k \geq 0$. This means that to analyze the middle product, we can exponentiate each diagonal block separately.

First, for $J_{\lambda_1} = J_r = [r]$ (a 1×1 matrix since r is a simple eigenvalue), we obtain:

$$\frac{1}{n_0 + m \cdot r} \cdot \prod_{k=0}^{m-1} 1 + \frac{1}{n_0 + k \cdot r} \cdot r = \frac{1}{n_0 + m \cdot r} \cdot \frac{n_0 + m \cdot r}{n_0} = \frac{1}{n_0}.$$

Now, for $\lambda \neq \lambda_1$: Our goal is to show that $\frac{1}{n_0+m\cdot r} \cdot \prod_{k=0}^{m-1} \left(I + \frac{1}{n_0+k\cdot r} \cdot J \right)$ converges to the zero matrix as m tends to infinity. For this it suffices to prove its operator norm approaches 0.

First, let γ denote $\left| \prod_{k=0}^{\lceil |\lambda| \rceil - 1} \left(I + \frac{1}{n_0+k\cdot r} \cdot J \right) \right|$, and observe this is a quantity depending only on n_0, r, λ and j, i.e. it is independent of m. Thus:

$$\left| \frac{1}{n_0 + m \cdot r} \cdot \prod_{k=0}^{m-1} \left(I + \frac{1}{n_0 + k \cdot r} \cdot J \right) \right| = \gamma \cdot \left| \frac{1}{n_0 + m \cdot r} \cdot \prod_{k=\lceil |\lambda| \rceil}^{m-1} \left(I + \frac{1}{n_0 + k \cdot r} \cdot J \right) \right|.$$

It therefore suffices to show that $\left| \frac{1}{n_0+m\cdot r} \cdot \prod_{k=\lceil |\lambda| \rceil}^{m-1} \left(I + \frac{1}{n_0+k\cdot r} \cdot J \right) \right|$ converges to 0 to conclude that the norm converges to 0. Note that for every $k \geq \lceil |\lambda| \rceil$:

$$I + \frac{1}{n_0 + k \cdot r} \cdot J_\lambda = I + \frac{1}{n_0 + k \cdot r} \cdot (\lambda \cdot I + N_\lambda) = I + \frac{\lambda}{n_0 + k \cdot r} \cdot I + \frac{1}{n_0 + k \cdot r} \cdot N_\lambda$$
$$= \left(1 + \frac{\lambda}{n_0 + k \cdot r} \right) \cdot \left(I + \frac{N_\lambda}{n_0 + k \cdot r + \lambda} \right).$$

Thus:

$$\left| \frac{1}{n_0 + m \cdot r} \cdot \prod_{k=\lceil |\lambda| \rceil}^{m-1} \left(I + \frac{1}{n_0 + k \cdot r} \cdot J \right) \right| = \left| \frac{1}{n_0 + m \cdot r} \cdot \prod_{k=\lceil |\lambda| \rceil}^{m-1} \left(1 + \frac{\lambda}{n_0 + k \cdot r} \right) \cdot \left(I + \frac{N_\lambda}{n_0 + k \cdot r + \lambda} \right) \right|$$
$$= \left| \frac{1}{n_0 + m \cdot r} \cdot \prod_{k=\lceil |\lambda| \rceil}^{m-1} \left(1 + \frac{\lambda}{n_0 + k \cdot r} \right) \cdot \prod_{k=\lceil |\lambda| \rceil}^{m-1} \left(I + \frac{N_\lambda}{n_0 + k \cdot r + \lambda} \right) \right|$$
$$= \frac{1}{n_0 + m \cdot r} \cdot \underbrace{\left| \prod_{k=\lceil |\lambda| \rceil}^{m-1} \left(1 + \frac{\lambda}{n_0+k\cdot r} \right) \right|}_{\text{A}} \cdot \underbrace{\left| \prod_{k=\lceil |\lambda| \rceil}^{m-1} \left(I + \frac{N_\lambda}{n_0+k\cdot r+\lambda} \right) \right|}_{\text{B}}.$$

According to Lemma 4.12 and Lemma 4.13, for sufficiently large m we have: $A \leq m^{\frac{\mathcal{R}e(\lambda)}{r}} \cdot \alpha_{\lambda,n_0,r}$, and $B \leq C_{N_\lambda} \cdot (\ln(m))^d$ where $\alpha_{\lambda,n_0,r}$ and C_{N_λ} are independent of m.

Observe that

$$\lim_{m\to\infty} \frac{1}{n_0 + m \cdot r} \cdot C_{N_\lambda} \cdot (\ln(m))^d \cdot m^{\frac{\mathcal{R}e(\lambda)}{r}} \cdot \beta_{\lambda,n_0,r} = 0,$$

and thus applying the squeeze rule we can conclude $\lim_{m\to\infty} \frac{1}{n_0+m\cdot r} \cdot A \cdot B = 0$, as needed.

To summarize:

$$\lim_{m\to\infty}\left(\frac{1}{n_0+m\cdot r}\cdot\prod_{k=0}^{m-1}\left(I+\frac{1}{n_0+k\cdot r}\cdot A\right)\right)\cdot v_{n_0}$$

$$=S\cdot diag\left(\frac{1}{n_0}\cdot I,\mathbf{0},...,\mathbf{0}\right)\cdot S^{-1}\cdot v_{n_0}=\frac{1}{n_0}\cdot P_r^A\cdot v_{n_0}=P_r^A\left(\frac{v_{n_0}}{n_0}\right).$$

4.3 Characterizing the Average Block Size

As mentioned in Section 2.2, inserting $(-\infty)$ into the structure prior to the workload affords a neat recurrence. An additional assumption we make for the purpose of a cleaner analysis for even splitting is that the first batch has size exactly $\frac{B-1}{2}$. Under this additional assumption, v_n only ever has support on $S=d+\langle r\rangle$. We are thus only interested in the behavior of $\frac{(v_n)_S}{n}$ as n tends to infinity. We let z_n denote $(v_n)_S$.

Observe that the recurrence for z_n can be represented by:

$$\forall n\geq d: z_{n+r}=\left(I+\frac{1}{n}\cdot A\left(B,r\right)_S\right)\cdot z_n,$$

where $v_d=e_1$. Combining Theorem 4.14 and Lemma 4.11, we obtain

$$\lim_{m\to\infty}\frac{z_{n_0+m\cdot r}}{n_0+m\cdot r}=P_r^{A(B,r)_S}\left(\frac{z_{n_0}}{n_0}\right),$$

where $P_r^{A(B,r)_S}(\cdot)$ is the spectral projection onto the generalized eigenspace of $A(B,r)$ corresponding to r (for further reading on spectral projections, refer to [14]). According to Lemma 4.11, r is a simple eigenvalue of $A\left(B,r\right)_S$, and thus the spectral projection is a projection onto the eigenspace corresponding to r. Overall, we conclude that $\frac{z_{n_0+m\cdot r}}{n_0+m\cdot r}$ converges to some vector in the eigenspace corresponding to r. Lemma 4.15 uses this fact to characterize the average block size using any eigenvector of $A(B,r)_S$.

Lemma 4.15. *Let $w=(d,d+1,...,B)$, and let u be any eigenvector corresponding to eigenvalue r for $A\left(B,r\right)_S$. Then the expected average block size converges to at least $\frac{\langle u,w_S\rangle}{\langle u,\vec{1}\rangle}$, where $\langle\cdot\rangle$ denotes the standard inner product.*

Proof. Note that at any time step n the identity $n=\sum_{k\in S}k\cdot x_k^n$ holds as an invariant. Furthermore, the average size of a block at time step n, is $\frac{n}{X^n}$ where $X^n=\sum_{k\in S}x_k^n$. Observe:

$$\frac{\langle z_n,w_S\rangle}{\langle z_n,\vec{1}\rangle}=\frac{\sum_{k\in S}k\cdot\mathbb{E}\left(x_k^n\right)}{\sum_{k\in S}\mathbb{E}\left(x_k^n\right)}=\frac{\mathbb{E}\left(\sum_{k\in S}k\cdot x_k^n\right)}{\mathbb{E}\left(\sum_{k\in S}x_k^n\right)}=n\cdot\frac{1}{\mathbb{E}\left(\sum_{k\in S}x_k^n\right)}$$

$$\leq_{\text{Jensen}}n\cdot\mathbb{E}\left(\frac{1}{\sum_{k\in S}x_k^n}\right)=\mathbb{E}\left(\frac{n}{\sum_{k\in S}x_k^n}\right)=\mathbb{E}\left(\frac{n}{X^n}\right).$$

Thus, the average blocks size converges to:

$$\lim_{m\to\infty}\frac{\left\langle z_{n_0+m\cdot r},w_S\right\rangle}{\left\langle z_{n_0+m\cdot r},\vec{1}\right\rangle}=\lim_{m\to\infty}\frac{\left\langle\frac{z_{n_0+m\cdot r}}{n_0+m\cdot r},w_S\right\rangle}{\left\langle\frac{z_{n_0+m\cdot r}}{n_0+m\cdot r},\vec{1}\right\rangle}.$$

We know by the preceding discussion in Section 4.3 that $\frac{z_{n_0+m\cdot r}}{n_0+m\cdot r}$ converges to some vector in the eigenspace of $A(B,r)_S$ corresponding to r. Denote this vector by u', and we thus have:

$$\lim_{m\to\infty} \frac{\left\langle \frac{z_{n_0+m\cdot r}}{n_0+m\cdot r}, w_S \right\rangle}{\left\langle \frac{z_{n_0+m\cdot r}}{n_0+m\cdot r}, \vec{1} \right\rangle} = \frac{\langle u', w_S\rangle}{\langle u', \vec{1}\rangle}.$$

Since r is a simple eigenvalue, the eigenspace corresponding to it is one-dimensional. This means there exists some scalar c such that $u' = c \cdot u$, therefore the limit equals:

$$\frac{\langle c\cdot u, w_S\rangle}{\langle c\cdot u, \vec{1}\rangle} = \frac{\langle u, w_S\rangle}{\langle u, \vec{1}\rangle} = \frac{\sum_{k\in S} k\cdot u_k}{\sum_{k\in S} u_k}.$$

□

Our goal is now to characterize an arbitrary eigenvector in the eigenspace corresponding to r to in order to provide a lower bound for the latter expression.

4.4 Lower Bounding the Average Block Size

We have established that the average block size converges to $\frac{\sum_{k\in S} k\cdot u_k}{\sum_{k\in S} u_k}$ for any eigenvector u of $A(B,r)_S$. Let u be a positive eigenvector of $A(B,r)_S$.

Lemma 4.16. *For every $k \in S$ such that $k \geq d + r$, we have $u_k = \frac{k-r}{k+r} \cdot u_{k-r}$.*

Proof. Let $k \in S$ such that $k \geq d+r$, let $\hat{u}$ be the expansion of u into a d-dimensional vector such that $\forall k \in S : \hat{u}_k = u_k$, and in all other position $\hat{u}$ is 0. Observe that $A_{k,*}\cdot\hat{u} = (A(B,r)_S)_{k,*}\cdot u$ where $A_{k,*}$ denotes the kth row of A. Observe that for every such k, $A_{k,k} = -k, A_{k,k-r} = k-r$ and all other entries are 0. Therefore, $(A(B,r)_S)_{k,*}\cdot u = A_{k,*}\cdot\hat{u} = -k\cdot u_k + (k-r)\,u_{k-r}$. Since u is an eigenvector of $A(B,r)_S$ we know that $(A(B,r)_S)_{k,*}\cdot u = r\cdot u_k$, so overall: $r\cdot u_k = -k\cdot u_k + (k-r)\,u_{k-r}$ and thus $u_k = \frac{k-r}{k+r}\cdot u_{k-r}$ holds for every $k \in S$ such that $k \geq d+r$. □

Now observe that it suffices to lower bound the term of interest, i.e. $\frac{\sum_{k\in S} k\cdot u_k}{\sum_{k\in S} u_k}$, when restricting the sums to some class in a partition of S, as formulated by the following lemma:

Lemma 4.17. *Let $C_1, ..., C_\ell$ be some partition of S. Then*

$$\frac{\sum_{k\in S} k\cdot u_k}{\sum_{k\in S} u_k} \geq min\left\{\frac{\sum_{k\in C_i} k\cdot u_k}{\sum_{k\in C_i} u_k} : i = 1, ..., \ell\right\}.$$

Proof. Note:

$$\frac{\sum_{k\in S} k\cdot u_k}{\sum_{k\in S} u_k} = \sum_{C_j} \lambda_j \cdot \frac{\sum_{k\in C_j} k\cdot u_k}{\sum_{k\in C_j} u_k},$$

where $\lambda_j = \frac{\sum_{k\in C_j} u_k}{\sum_{k\in S} u_k}$. Now observe that $\sum_{C_j}\lambda_j = 1$, and thus $\frac{\sum_{k\in S} k\cdot u_k}{\sum_{k\in S} u_k}$ is a weighted average of $\left\{\frac{\sum_{k\in C_1} k\cdot u_k}{\sum_{k\in C_1} u_k}, ..., \frac{\sum_{k\in C_\ell} k\cdot u_k}{\sum_{k\in C_\ell} u_k}\right\}$, and is therefore at least the minimum term. □

In the partition of S we will use, each class C will be of the form $\{j_0, j_1, ..., j_L\}$ where $d \leq j_0 \leq d+r-1$, $\forall i : j_i = j_0 + r\cdot i$, and L is the maximum integer for which $j_0 + L\cdot i \leq B$. According to Lemma 4.17, it suffices to set such a class C and provide a lower bound for $\frac{\sum_{i=0}^{L} j_i\cdot u_{j_i}}{\sum_{i=0}^{L} u_{j_i}}$. By Lemma 4.16, $u_{j_i} = \frac{j_{i-1}}{j_{i+1}}\cdot u_{j_{i-1}}$ for every $1 \leq i \leq L$, letting j_{L+1} denote $j_L + r$. Thus, for every $\ell \geq 2$:

$$u_{j_\ell} = u_{j_0}\cdot\prod_{i=1}^{\ell}\frac{j_{i-1}}{j_{i+1}} = u_{j_0}\cdot\frac{j_0\cdot j_1}{j_\ell\cdot j_{\ell+1}}.$$

Observe that the above equality also holds for $\ell = 0, 1$. Overall we get: for every $\ell \geq 0$: $u_{j_\ell} = \frac{j_0 \cdot (j_0+r)}{j_\ell \cdot (j_\ell+r)} \cdot u_{j_0}$. So:

$$\frac{\sum_{i=0}^{L} j_i \cdot u_{j_i}}{\sum_{i=0}^{L} u_{j_i}} = \frac{\sum_{i=0}^{L} j_i \cdot \frac{j_0 \cdot (j_0+r)}{j_i \cdot (j_i+r)} \cdot u_{j_0}}{\sum_{i=0}^{L} \frac{j_0 \cdot (j_0+r)}{j_i \cdot (j_i+r)} \cdot u_{j_0}}$$

$$= \frac{j_0 \cdot (j_0+r) \cdot u_{j_0} \cdot \sum_{i=0}^{L} \frac{1}{j_i+r}}{j_0 \cdot (j_0+r) \cdot u_{j_0} \cdot \sum_{i=0}^{L} \frac{1}{j_i \cdot (j_i+r)}} = \frac{\sum_{i=0}^{L} \frac{1}{j_i+r}}{\sum_{i=0}^{L} \frac{1}{j_i \cdot (j_i+r)}}. \tag{9}$$

We now provide three lower bounds for the average block size that reflect the different parameter regimes for r. The first lower bound is favorable to the others when $r < 0.0058B$, the second lower bound is favorable when $r < 0.21B$, and the third lower bound is favorable in the rest of the regime we analyze for even splitting, i.e. $0.21B \leq r < 0.5B$. The first two lower bounds are obtained by lower bounding the right hand side of Equation (9). Then, we consider a refinement to the partition we described for S in order to obtain a third lower bound.

4.4.1 First lower bound - small values of r. Our goal in this part is to prove that for every $r \leq 0.0058B$ we have:

$$\frac{\sum_{i=0}^{L} \frac{1}{j_i+r}}{\sum_{i=0}^{L} \frac{1}{j_i \cdot (j_i+r)}} \geq B \left(\ln(2) - \frac{5r}{B} \right).$$

Begin by observing:

$$\sum_{i=0}^{L} \frac{1}{j_i \, (j_i + r)} = \frac{1}{r} \cdot \left(\frac{1}{j_0} - \frac{1}{j_{L+1}} \right).$$

Thus:

$$\frac{\sum_{i=0}^{L} \frac{1}{j_i+r}}{\sum_{i=0}^{L} \frac{1}{j_i \cdot (j_i+r)}} = \frac{r \cdot \sum_{i=0}^{L} \frac{1}{j_i+r}}{\frac{1}{j_0} - \frac{1}{j_{L+1}}} \geq \frac{r \cdot \sum_{i=0}^{L} \frac{1}{j_i+r}}{\frac{1}{d} - \frac{1}{B+r}}.$$

Now note:

$$\sum_{i=0}^{L} \frac{1}{j_i + r} = \sum_{i=1}^{L+1} \frac{1}{j_0 + i \cdot r} \geq \int_{j_0+r}^{j_0+(L+1)\cdot r} \frac{dx}{x} \geq \ln(B+1) - \ln(j_0 + r) \geq \frac{1}{r} \cdot \ln \left(\frac{B+1}{d+2r} \right).$$

So:

$$\frac{\sum_{i=0}^{L} \frac{1}{j_i+r}}{\sum_{i=0}^{L} \frac{1}{j_i \cdot (j_i+r)}} \geq \frac{r \cdot \sum_{i=0}^{L} \frac{1}{j_i+r}}{\frac{1}{d} - \frac{1}{B+r}} \geq \frac{\ln \left(\frac{B+1}{d+2r} \right)}{\frac{1}{d} - \frac{1}{B+r}} \geq \frac{\ln \left(\frac{2(B+1)}{B+1+4r} \right)}{\frac{2}{B+1} - \frac{1}{B+r}} = \ln \left(\frac{2}{1 + \frac{4r}{B+1}} \right) \cdot \frac{(B+1) \cdot (B+r)}{B + 2r - 1}.$$

Observe that $\frac{(B+1)\cdot(B+r)}{B+2r-1} \geq B \cdot \left(1 - \frac{r}{B}\right)$, and $\ln \left(\frac{2}{1+\frac{4r}{B+1}} \right) \geq \ln(2) - \frac{4r}{B+1}$ (as $\ln \left(\frac{2}{1+x} \right) \geq \ln(2) - x$ for every $x \geq 0$). So:

$$\frac{\sum_{i=0}^{L} \frac{1}{j_i+r}}{\sum_{i=0}^{L} \frac{1}{j_i \cdot (j_i+r)}} \geq B \cdot \left(\ln(2) - \frac{4r}{B+1} \right) \cdot \left(1 - \frac{r}{B} \right) \geq B \cdot \left(\ln(2) - \frac{r \cdot (4 + \ln(2))}{B} \right) \geq B \cdot \left(\ln(2) - \frac{5r}{B} \right).$$

4.4.2 Second lower bound - moderate values of r. Our goal in this part is to prove that for every $r \leq 0.21B$ we have:

$$\frac{\sum_{i=0}^{L} \frac{1}{j_i+r}}{\sum_{i=0}^{L} \frac{1}{j_i \cdot (j_i+r)}} \geq \frac{2B \cdot (B+1)}{3B+1+2r}.$$

Begin by observing:

$$\sum_{i=0}^{L} \frac{1}{j_i \, (j_i+r)} = \frac{1}{r} \cdot \left(\frac{1}{j_0} - \frac{1}{j_{L+1}} \right) = \frac{j_{L+1} - j_0}{r \cdot j_0 \cdot j_{L+1}}.$$

Thus:

$$\frac{\sum_{i=0}^{L} \frac{1}{j_i+r}}{\sum_{i=0}^{L} \frac{1}{j_i \cdot (j_i+r)}} = r \cdot \frac{j_0 \cdot j_{L+1}}{j_{L+1} - j_0} \cdot \sum_{i=0}^{L} \frac{1}{j_i + r}$$

$$= r \cdot \frac{j_0 \cdot j_{L+1}}{j_0 + r \cdot (L+1) - j_0} \cdot \sum_{i=0}^{L} \frac{1}{j_i + r} = j_0 \cdot j_{L+1} \cdot \left(\frac{1}{L+1} \cdot \sum_{i=1}^{L+1} \frac{1}{j_i} \right).$$

According to Jensen's inequality the average of reciprocals is at least the reciprocal of the average, and observe: $\frac{1}{L+1} \cdot \sum_{i=1}^{L+1} j_i = \frac{j_1 + j_{L+1}}{2}$. Thus:

$$j_0 \cdot j_{L+1} \cdot \left(\frac{1}{L+1} \cdot \sum_{i=1}^{L+1} \frac{1}{j_i} \right) \geq j_0 \cdot j_{L+1} \cdot \frac{2}{j_1 + j_{L+1}} = \frac{2 \cdot j_0 \cdot j_{L+1}}{j_0 + j_{L+1} + r}.$$

Observe that the two dimensional function $f(x, y) = \frac{2 \cdot x \cdot y}{x+y+r}$ is increasing in both x, y for $x > 0, y > 0$, since both partial derivatives are positive. Thus, if we observe the function in an axis-aligned rectangle contained in the first quadrant with bottom left corner (x_0, y_0), the function is minimized in (x_0, y_0). Observe the function in the axis-aligned rectangle given by $d \leq x \leq d + r - 1$ and $B \leq y \leq B + 2r$; then $f(x, y)$ is minimized at $\frac{2 \cdot d \cdot B}{d+B+r} = \frac{2(B+1) \cdot B}{3B+1+2r}$. Note that (j_0, j_{L+1}) is within this rectangle and therefore:

$$\frac{2 \cdot j_0 j_{L+1}}{j_0 + j_{L+1} + r} \geq \frac{2 \, (B+1) \cdot B}{3B+1+2r}.$$

4.4.3 Third lower bound - larger values of r. Our goal in this section is to show that for every $1 \leq r \leq \frac{5}{12}B$, the average block size converges to a value that is at least $\frac{7}{12}B$. [5] Towards this end, we consider a refinement of the partition on S we considered in Section 4.3. Recall each class C is of the form $\{j_0, j_1, ..., j_\ell\}$ where $d \leq j_0 \leq d + r - 1$ and for every i, $j_i = j_0 + r \cdot i$. Furthermore, recall that

$$\forall \ell \geq 0 : u_{j_\ell} = \frac{j_0 \cdot (j_0 + r)}{j_\ell \cdot (j_\ell + r)} \cdot u_{j_0} \coloneqq \frac{1}{j_\ell \cdot (j_\ell + r)} \cdot \kappa_C. \quad (\star)$$

Our refinement is obtained by further partitioning each such class C into

$$\{\{j_i, j_{L-i}\} : i \in \{0, 1, ..., \lfloor L/2 \rfloor\}\}.$$

According to Lemma 4.17, it suffices to provide a lower bound on

$$F(a, b, r) = \frac{a \cdot u_a + b \cdot u_b}{u_a + u_b},$$

[5] We note that as r grows, some of the classes we define may become singletons. But this only occurs if $j_0 + r \geq B \iff j_0 \geq B - r$. Under the assumption that $r \leq \frac{5}{12}B$ this would imply $j_0 \geq \frac{7}{12}B$. So, the contribution of such a class for the weighted average without any further refinement is $\frac{\sum_{i=0}^{L} \frac{1}{j_i+r}}{\sum_{i=0}^{L} \frac{1}{j_i \cdot (j_i+r)}} = j_0 \geq \frac{7}{12}B$, respecting the stated bound. The refinement is used to argue about the remaining classes.

where $a = j_i$ and $b = j_{L-i}$ for some $i \in \{0, ..., \lfloor L/2 \rfloor\}$. Plugging in $(\star)$ we obtain:

$$F(a,b,r) = \frac{a \cdot \frac{\kappa_C}{a(a+r)} + b \cdot \frac{\kappa_C}{b(b+r)}}{\frac{\kappa_C}{a(a+r)} + \frac{\kappa_C}{b(b+r)}}, = \frac{\frac{1}{a+r} + \frac{1}{b+r}}{\frac{1}{a(a+r)} + \frac{1}{b(b+r)}} = \frac{ab(a+b+2r)}{a^2+b^2+r(a+b)}.$$

Lemma 4.18. *$F(a,b,r)$ is minimized at $a = j_0$ and $b = j_L$.*

Proof. Fixing $r > 0$ and $s > 0$, we can show that

$$F(a, s-a, r) = \frac{a(s-a)(s+2r)}{a^2 + (s-a)^2 + rs}$$

is strictly increasing in a for $a \in (0, s/2)$ by taking a partial derivative on a. So, fixing r and $s = j_0 + j_L$, we have for every $i \in \{0, ..., \lfloor L/2 \rfloor\}$,

$$F(j_i, (j_0 + j_L) - j_i, r) \geq F(j_0, j_L, r).$$

Observe that $(j_0 + j_L) - j_i = j_{L-i}$ and thus for every $i \in \{0, ..., \lfloor L/2 \rfloor\}$,

$$F(j_i, j_{L-i}, r) \geq F(j_0, j_L, r).$$

□

To complete the analysis, we set $x \coloneqq \frac{j_0}{B}$,$y \coloneqq \frac{j_L}{B}$, and $\alpha \coloneqq \frac{r}{B}$. Then, we define:

$$f(x,y,\alpha) \coloneqq \frac{F(j_0, j_L, r)}{B} = \frac{xy(x+y+2\alpha)}{x^2+y^2+\alpha(x+y)}.$$

What's left to prove now is that $f(x,y,\alpha) \geq \frac{7}{12}$ in the feasible domain, i.e:

$$\frac{1}{2} \leq x \leq 1, \quad 1-\alpha < y < 1, \quad y - x \geq \alpha, \quad 0 < \alpha < \frac{1}{2)}.$$

This will imply the average block size is at least $\frac{7}{12} \cdot B$, as needed.

Lemma 4.19. *Let*

$$f(x,y,\alpha) \coloneqq \frac{xy(x+y+2\alpha)}{x^2+y^2+\alpha(x+y)}.$$

Assume $\frac{1}{2} \leq x \leq 1$, $0 < \alpha < \frac{1}{2}$, $1 - \alpha \leq y \leq 1$, and $y - x \geq \alpha$. Then $f(x,y,\alpha) \geq \frac{7}{12}$.

Proof. We begin by showing f is monotonic in y in the stated regime.

$$\frac{\partial f}{\partial y} = \frac{x(\alpha+x)\left(2\alpha x + x^2 + 2xy - y^2\right)}{\left(x^2+y^2+\alpha(x+y)\right)^2}.$$

The sign of the derivative is thus determined by $2\alpha x + x^2 + 2xy - y^2$. Observe:

$$2\alpha x + x^2 + 2xy - y^2 = 2\alpha x + x^2\left(1 + 2 \cdot \frac{y}{x} - \left(\frac{y}{x}\right)^2\right).$$

Now, since $\frac{1}{2} \leq x < y < 1$, we have $\frac{y}{x} \in [1,2]$. Observe that $1 + 2t - t^2 \geq 1$ on $[1,2]$. Thus:

$$2\alpha x + x^2\left(1 + 2 \cdot \frac{y}{x} - \left(\frac{y}{x}\right)^2\right) \geq 2\alpha x + x^2 > 0.$$

So, f is strictly increasing in y in the stated regime. Now, given our constraints, i.e. $y \geq 1 - \alpha$,$y \geq x + \alpha$, $y \geq 1 - \alpha$, and thus y is minimized at $y_{\min} \coloneqq \max\{x + \alpha, 1 - \alpha\}$. Therefore f is minimized at $f(x, y_{\min}, \alpha)$, and it is sufficient to show this value is at least $7/12$. We divide into two cases:

Case I. $x \geq 1-2\alpha$*:* In this case $y_{\min} = x+\alpha$. Observe that since $x \leq y-\alpha$ and $y<1$, we know that $x < 1-\alpha$. Overall we have:

$$f(x, y_{\min}, \alpha) = f(x, x+\alpha, \alpha) = \frac{x(3\alpha+2x)}{2(x+\alpha)} =: g(x,\alpha),$$

with

$$x \in \left[\max\{1/2, 1-2\alpha\}, 1-\alpha\right], \qquad 0<\alpha<1/2.$$

Now, if $\alpha > \frac{1}{4}$, then $x \in \left[\frac{1}{2}, 1-\alpha\right]$. Computing partial derivatives reveals that g is increasing in both x and α, as long as $x, \alpha > 0$, and thus $g(x,\alpha) \geq g\left(\frac{1}{2}, \frac{1}{4}\right) = \frac{7}{12}$. Otherwise, $\alpha \leq \frac{1}{4}$, and then $x \in [1-2\alpha, 1-\alpha]$, and since g is increasing in x: $g(x,\alpha) \geq g(1-2\alpha,\alpha) = -\frac{(\alpha-2)(2\alpha-1)}{2\alpha-2}$. Observe that $-\frac{(\alpha-2)(2\alpha-1)}{2\alpha-2} \geq \frac{7}{12}$ for every $\alpha \leq \frac{1}{4}$ (equality occurs at $\alpha = \frac{1}{4}$).

Case II. $x < 1-2\alpha$*:* In this case $y_{\min} = 1-\alpha$, and thus

$$f(x, y_{\min}, \alpha) = f(x, 1-\alpha, \alpha) \coloneqq h(x,\alpha) = \frac{x(1-\alpha)(x+1+\alpha)}{x^2+1-\alpha+\alpha x}.$$

Taking a partial derivative on h reveals it is increasing in x for $x \in \left[\frac{1}{2}, 1-2\alpha\right]$, and thus $h(x,\alpha) \geq h(\frac{1}{2},\alpha) \coloneqq \varphi(\alpha) = \frac{2\alpha^2+\alpha-3}{2\alpha-5}$. Now, one can easily verify that $\varphi(\alpha) \geq \frac{7}{12}$ (equality occurs at $\alpha = \frac{1}{4}$). □

5 Algorithms for Larger Batch Sizes

Our analysis for even splitting shows that as long as $r \leq \left(\frac{1}{2}-\varepsilon\right)\cdot B$ for some constant $\varepsilon > 0$, even splitting will produce fullness of at least $\frac{1}{2}+\delta$ for a constant $\delta > 0$. We construct algorithms that can beat the trivial 0.5 fullness bound for larger values of r, with different algorithms for different regimes. Note that these algorithms are 'r aware' in the sense that r is used explicitly in the algorithm's description. We see that large batch sizes admit simple algorithms, which are far easier to analyze. Refer to Section 3.1 for a high-level overview.

5.1 Uneven Splitting for Moderately Large Batches

We present here a subroutine that is a generic description of how our uneven splitting algorithms handle splits that occur during batches of insertions. The subroutine receives as input a memory block A containing B keys in sorted order, i.e. a full memory block, a key k to be inserted into A, and targets f_L,f_R such that $f_L + f_R > B$. The goal of the subroutine is to split the block, insert k, and then handle the following insertions in the batch such that the two memory blocks created as result of the split, A_L and A_R, contain f_L and f_R keys.

Subroutine 1 TargetSplit$((A = k_1 < \ldots < k_B), k, f_L, f_R)$

Let $j = |\{k_i : i \in [B], k_i < k\}|$
if $j \geq f_L$ **then**
    Split A into $A_L = k_1, \ldots, k_{f_L}$ and $A_R = k_{f_L+1}, \ldots, k_B$.
    Insert k and the following $f_L + f_R - B - 1$ keys in the batch into A_R.
else if $j \leq B - f_R$ **then**
    Split A into $A_L = k_1, \ldots, k_{B-f_R}$ and $A_R = k_{B-f_R+1}, \ldots, k_B$.
    Insert k and the following $f_L + f_R - B - 1$ keys in the batch into A_L.
else
    Split A into $A_L = k_1, \ldots, k_j$ and $A_R = k_{j+1}, \ldots, k_B$.
    Insert k and the following $f_L - j - 1$ keys in the batch into A_L.
    After that, insert the next $f_R + j - B$ keys in the batch into A_R.

Using this subroutine, we obtain Lemma 5.1 and Lemma 5.2. More specifically, for $\frac{B}{3} < r \le \frac{B}{2}$, we call TargetSplit$(A, k, r, 2r)$ when blocks become full. For $\frac{2B}{5} < r \le \frac{2B}{3}$, we call TargetSplit$(A, k, \frac{r}{2}, \frac{3r}{2})$ or TargetSplit$(A, k, r, \frac{3r}{2})$ when a block becomes full, depending on its size prior to the batch. For more details, refer to the full version of the paper. [6]

Lemma 5.1. *When $\frac{B}{3} < r \le \frac{B}{2}$, the average fullness obtained by uneven splitting is at least $\frac{3r}{2B}$.*

Lemma 5.2. *When $\frac{2B}{5} < r \le \frac{2B}{3}$, the average fullness obtained by uneven splitting is at least $\frac{10r}{9B}$.*

5.2 Deferred Even Splitting for Large Batches

When $r > \frac{2}{3}B$, we obtain good space utilization by using deferred even splitting, as defined in Section 1. First, note that for $\frac{2}{3}B < r \le B$, deferred even splitting is trivial to analyze—at the end of each batch all blocks invariably have size exactly r. Thus we obtain the following result.

Lemma 5.3. *When $\frac{2}{3}B < r \le B$, the average fullness obtained by deferred even splitting is at least $\frac{r}{B}$.*

For deferred even splitting when $r > B$, we obtain the following lemma, which proof can be found in the full version of the paper.[6]

Lemma 5.4. *When $r > \frac{2}{3}B$, the average fullness obtained by deferred even splitting is at least* $\max\left\{\frac{2}{3}, \frac{r/B+\frac{1}{2}}{\lceil r/B+1\rceil}\right\} - \frac{1}{B}$.

6 Deferred Even Splitting for Smaller Batches

In this section, we present a closed form for the limiting fullness of the deferred even split strategy for certain values of the batch size r.

Recall that under deferred even split we always insert in batches of size r, and we only split a block when a single insertion causes it to exceed capacity B. When $r \in (B/(2i), B/(2i-1)]$, every block size is forced to be a multiple of r, and the largest possible multiple is $(2i-1)r$. After rescaling by a factor of r, the process on block sizes becomes exactly Yao's even-split model with capacity $2i-1$ and split rule $2i \to (i, i)$. This allows us to read off a closed form for the stationary distribution and hence for the expected fullness. We obtain the following lemma, which proof can be found in the full version of the paper.[6]

Lemma 6.1. *Suppose B and i are positive integers. For $r \in \left(B/(2i),\ B/(2i-1)\right]$, the expected final fullness of the deferred even split strategy is*

$$\frac{2ir}{B}\,(H_{2i} - H_i),$$

where $H_k = 1 + \frac{1}{2} + \cdots + \frac{1}{k}$ is the k-th harmonic number.

Acknowledgments

This work was supported by NSF grants CNS-2504471 and CCF-2247577. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Michael Bender was supported in part by the John L. Hennessy Chaired Professorship and Sandia National Laboratories. Aaron Bernstein is funded by the NSF CAREER grant, the Sloan fellowship, and the Charles S. Baylis endowment at NYU. Martín Farach-Colton was supported in part by the Leonard J. Shustek professorship. Hanna Komlós was supported in part by the Graduate Fellowships for STEM Diversity. Part of this work was supported by the Simons Institute for the Theory of Computing, and conducted when Hanna Komlós was visiting the Institute.

[6] https://arxiv.org/abs/2603.12211

References

[1] [n. d.]. Behind the Scenes on OPTIMIZE_FOR_SEQUENTIAL_KEY | Microsoft Community Hub — techcommunity.microsoft.com. https://techcommunity.microsoft.com/blog/sqlserver/behind-the-scenes-on-optimize-for-sequential-key/806888?utm_source=chatgpt.com. [Accessed 08-12-2025].

[2] [n. d.]. How to Check Fragmentation in an Oracle Database — dev.to. https://dev.to/dm8ry/how-to-check-fragmentation-in-an-oracle-database-3j7d?utm_source=chatgpt.com. [Accessed 07-12-2025].

[3] [n. d.]. Understanding CRUD Operations on Tables with B-tree Indexes, Page-splits, and Fragmentation – SQLServerCentral — sqlservercentral.com. https://www.sqlservercentral.com/articles/understanding-curd-operations-on-tables-with-b-tree-indexes-page-splits-and-fragmentation?utm_source=chatgpt.com. [Accessed 07-12-2025].

[4] Daniar Achakeev and Bernhard Seeger. 2013. Efficient Bulk Updates on Multiversion B-trees. *Proc. VLDB Endow.* 6, 14 (2013), 1834–1845.

[5] Adnan Alhomssi and Viktor Leis. 2021. Contention and Space Management in B-Trees. In *CIDR*. www.cidrdb.org.

[6] Widyastuti Andriyani and Pujianto Pujianto. 2024. Maintaining Query Performance through Table Rebuilding & Archiving. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)* 18, 1 (2024), 73–82.

[7] Ricardo A. Baeza-Yates. 1995. Fringe Analysis Revisited. *ACM Comput. Surv.* 27, 1 (1995), 111–119.

[8] Hokeun Cha, Xiangpeng Hao, Tianzheng Wang, Huanchen Zhang, Aditya Akella, and Xiangyao Yu. 2023. Blink-hash: An Adaptive Hybrid Index for In-Memory Time-Series Databases. *Proc. VLDB Endow.* 16, 6 (2023), 1235–1248.

[9] Siying Dong, Mark Callaghan, Leonidas Galanis, Dhruba Borthakur, Tony Savor, and Michael Strum. 2017. Optimizing Space Amplification in RocksDB. In *CIDR*. www.cidrdb.org.

[10] Paul DuBois. 2013. *MySQL*. Addison-Wesley.

[11] Bernhard Eisenbarth, Nivio Ziviani, Gaston H. Gonnet, Kurt Mehlhorn, and Derick Wood. 1982. The Theory of Fringe Analysis and Its Application to 2-3 Trees and B-Trees. *Inf. Control.* 55, 1-3 (1982), 125–174.

[12] Grant Fritchey. 2012. Fragmentation analysis. In *SQL Server 2012 Query Performance Tuning*. Springer, 211–239.

[13] Goetz Graefe. 2024. More Modern B-Tree Techniques. *Found. Trends Databases* 13, 3 (2024), 169–249.

[14] Roger A. Horn and Charles R. Johnson. 1990. *Matrix Analysis*. Cambridge University Press. http://www.amazon.com/Matrix-Analysis-Roger-Horn/dp/0521386322%3FSubscriptionId%3D192BW6DQ43CK9FN0ZGG2%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0521386322

[15] Ram Kesavan, Matthew Curtis-Maury, Vinay Devadas, and Kesari Mishra. 2020. Countering Fragmentation in an Enterprise Storage System. *ACM Trans. Storage* 15, 4 (2020), 25:1–25:35.

[16] Sang-Wook Kim. 2002. On batch-constructing B+-trees: algorithm and its performance evaluation. *Inf. Sci.* 144, 1-4 (2002), 151–167.

[17] Roland Kühn, Daniel Biebert, Christian Hakert, Jian-Jia Chen, and Jens Teubner. 2023. Towards Data-Based Cache Optimization of B+-Trees. In *DaMoN*. ACM, 63–69.

[18] Yongsik Kwon, Seonho Lee, Yehyun Nam, Joong Chae Na, Kunsoo Park, Sang K. Cha, and Bongki Moon. 2023. DB+-tree: A new variant of B+-tree for main-memory database systems. *Inf. Syst.* 119 (2023), 102287.

[19] Percona LLC. [n. d.]. Percona Server for MySQL - TokuDB Introduction — docs.percona.com. https://docs.percona.com/percona-server/5.7/tokudb/tokudb_intro.html. [Accessed 08-12-2025].

[20] Raghu Ramakrishnan and Johannes Gehrke. 2003. *Database management systems (3. ed.)*. McGraw-Hill.

[21] Aneesh Raman, Subhadeep Sarkar, Matthaios Olma, and Manos Athanassoulis. 2022. OSM-tree: A Sortedness-Aware Index. *CoRR* abs/2202.04185 (2022).

[22] A. M. Tamhankar and S. Ram. 1998. Database fragmentation and allocation: an integrated methodology and case study. *IEEE Trans. Syst. Man Cybern. Part A* 28, 3 (1998), 288–305.

[23] Dimitrios Tsitsigkos, Achilleas Michalopoulos, Nikos Mamoulis, and Manolis Terrovitis. 2025. BS-tree: A gapped data-parallel B-tree. *CoRR* abs/2505.01180 (2025).

[24] Ziqi Wang, Andrew Pavlo, Hyeontaek Lim, Viktor Leis, Huanchen Zhang, Michael Kaminsky, and David G. Andersen. 2018. Building a Bw-Tree Takes More Than Just Buzz Words. In *SIGMOD Conference*. ACM, 473–488.

[25] Andrew Chi-Chih Yao. 1978. On Random 2-3 Trees. *Acta Informatica* 9 (1978), 159–170.

[26] Sepanta Zeighami and Raymond Chi-Wing Wong. 2020. Bridging the Gap Between Theory and Practice on Insertion-Intensive Database. *CoRR* abs/2003.01064 (2020).

Received December 2025; accepted February 2026