THEORY OF COMPUTING, Volume 19 (8), 2023, pp. 1–71
www.theoryofcomputing.org

SPECIAL ISSUE: APPROX-RANDOM 2020

Pinning Down the Strong Wilber-1 Bound for Binary Search Trees

Parinya Chalermsook[‡] Julia Chuzhoy[§] Thatchaphol Saranurak

Received June 13, 2021; Revised August 10, 2022; Published December 19, 2023

Abstract. Dynamic Optimality Conjecture, postulating the existence of an $O(1)$-competitive online algorithm for binary search trees (BSTs), is among the most fundamental open problems in dynamic data structures. The conjecture remains wide open, despite extensive work and some notable progress, including, for example, the $O(\log\log n)$-competitive Tango Trees, which is the best currently known competitive ratio. One of the main hurdles towards settling the conjecture is that we currently do not have polynomial-time approximation algorithms achieving better than an $O(\log\log n)$-approximation, even in the offline setting. All known non-trivial algorithms for BSTs rely on comparing the algorithm's cost with the so-called Wilber-1 bound (WB-1). Therefore, establishing the worst-case relationship

An extended abstract of this paper appeared in the Proceedings of the 23rd Internat. Conf. on Approximation Algorithms and Combinatorial Optimization (APPROX'20)

[‡]Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 759557) and by the Academy of Finland Research Fellows program, under grant No. 310415.

[§]Supported in part by NSF grant CCF-1616584. Part of the work was done while the second author was a Weston visiting professor at the Department of Computer Science and Applied Mathematics, Weizmann Institute of Science.

ACM Classification: Theory of computation → Data structure design and analysis

AMS Classification: 68Q25, 68W25

Key words and phrases: binary search trees, dynamic optimality, data structures

DOI: 10.4086/toc.2023.v019a008

between this bound and the optimal solution cost appears crucial for further progress, and it is an interesting open question in its own right.

Our contribution is twofold. First, we show that the gap between WB-1 and the optimal solution value can be as large as $\Omega(\log\log n/\log\log\log n)$; in fact, we show that the gap holds even for several stronger variants of the bound.[1] Second, we show, given an integer $D > 0$, a D-approximation algorithm that runs in time $\exp\left(O\left(n^{1/2^{\Omega(D)}}\log n\right)\right)$. In particular, this yields a constant-factor approximation algorithm with subexponential running time.[2] Moreover, we obtain a simpler and cleaner efficient $O(\log\log n)$-approximation algorithm that can be used in an online setting. Finally, we suggest a new bound, that we call the *Guillotine Bound*, that is stronger than WB-1, while maintaining its algorithm-friendly nature, that we hope will lead to better algorithms. All our results use the geometric interpretation of the problem, leading to cleaner and simpler analysis.

1 Introduction

1.1 Binary search trees

Binary search trees (BST's) are a fundamental data structure that has been extensively studied for many decades. Informally, suppose we are given as input an **online** access sequence $X = \{x_1, \ldots, x_m\}$ of keys from $\{1, \ldots, n\}$, and our goal is to maintain a binary search tree T over the set $\{1, \ldots, n\}$ of keys. The algorithm is allowed to modify the tree T after each access; the tree obtained after the ith access is denoted by T_{i+1}. Each such modification involves a sequence of *rotation* operations that transform the current tree T_i into a new tree T_{i+1}. The cost of the transformation is the total number of rotations performed plus the depth of the key x_i in the tree T_i. The total cost of the algorithm is the total cost of all transformations performed as the sequence X is processed. We denote by $\mathsf{OPT}(X)$ the smallest cost of any algorithm for maintaining a BST for the access sequence X, when the whole sequence X is known to the algorithm in advance.

Several algorithms for BST's, whose costs are guaranteed to be $O(m \log n)$ for any access sequence, such as AVL-trees [1] and red-black trees [2], are known since the 60's (see [10], Chapters 12 and 13). Moreover, it is well known that there are length-m access sequences X on n keys, for which $\mathsf{OPT}(X) = \Omega(m \log n)$. However, such optimal worst-case guarantees are often unsatisfactory from both practical and theoretical perspectives, as one can often obtain better results for "structured" inputs. Arguably, a better notion of the algorithm's performance to consider is *instance optimality*, where the algorithm's performance is compared to the optimal cost $\mathsf{OPT}(X)$ for the specific input access sequence X. This notion is naturally captured by the algorithm's *competitive ratio*: we say that an algorithm for BST's is *α-competitive*, if, for every

[1]A recent independent paper by Lecomte and Weinstein (ESA'20) shows an even stronger, $\Omega(\log\log n)$, separation.

[2]The term "subexponential time" in this paper refers to the running time $2^{o(n)}$.

online input access sequence X, the cost of the algorithm's execution on X is at most $\alpha \cdot \mathsf{OPT}(X)$. Since for every length-m access sequence X, $\mathsf{OPT}(X) \geq m$, the above-mentioned algorithms that provide worst-case $O(m \log n)$-cost guarantees are also $O(\log n)$-competitive. However, there are many known important special cases, in which the value of the optimal solution is $O(m)$, and for which the existence of an $O(1)$-competitive algorithm would lead to a much better performance, including some interesting applications, such as, for example, adaptive sorting [27, 7, 23, 26, 15, 24, 13, 9, 8, 3, 6, 5, 19].

1.1.1 The Dynamic Optimality Conjecture

A striking conjecture of Sleator and Tarjan [25] from 1985, called the *dynamic optimality conjecture*, asserts that the *Splay Trees* provide an $O(1)$-competitive algorithm for BST's. This conjecture has sparked a long line of research, but despite the continuing effort, and the seeming simplicity of BST's, it remains widely open. In a breakthrough result, Demaine et al. [12] proposed the Tango Trees algorithm, that achieves an $O(\log \log n)$-competitive ratio, and has remained the best known algorithm for the problem, for over 15 years. A natural avenue for overcoming this barrier is to first consider the "easier" task of designing (offline) approximation algorithms, whose approximation factor is below $O(\log \log n)$. Designing better approximation algorithms is often a precursor to obtaining better online algorithms, and it is a natural stepping stone towards this goal.

1.1.2 The Wilber bounds

The main obstacle towards designing better algorithms, both in the online and the offline settings, is obtaining tight lower bounds on the value $\mathsf{OPT}(X)$, that can be used in algorithm design. In order to improve upon the trivial $O(\log n)$ approximation, the lower bound $\mathsf{OPT}(X) \geq m$ is not sufficient[1]. Wilber [29] proposed two new bounds, that we refer to as the Wilber-1 Bound, or Wilber's first bound, (WB-1) and the Wilber-2 Bound (WB-2). He proved that, for every input sequence X, the values of both these bounds on X are at most $\mathsf{OPT}(X)$. The breakthrough result of Demaine et al. [12], that gives an $O(\log \log n)$-competitive online algorithm, relies on the WB-1 bound. In particular, they show that the cost of the solution produced by their algorithm is within an $O(\log \log n)$-factor from the WB-1 bound on the given input sequence X, and hence from $\mathsf{OPT}(X)$. This in turn implies that, for every input sequence X, the value of the WB-1 bound is within an $O(\log \log n)$ factor from $\mathsf{OPT}(X)$. Follow-up work [28, 16] improved several aspects of Tango Trees, but it did not improve the approximation factor. Additional lower bounds on OPT, that subsume both the WB-1 and the WB-2 bounds, were suggested in [11, 14, 17], but unfortunately it is not clear how to exploit them in algorithm design. To this day, the only method we have for designing non-trivial online or offline approximation algorithms for BST's is by relying on the WB-1 bound, and this seems to be the most promising approach for obtaining better algorithms. In order to make further progress on both online and offline

[1] This is due to the existence of sequences X with $\mathsf{OPT}(X) = \Omega(m \log n)$.

approximation algorithms for BST's, it therefore appears crucial that we better understand the relationship between the WB-1 bound and the optimal solution cost.

Informally, the WB-1 bound relies on recursive partitioning of the input key sequence, that can be represented by a partitioning tree. The standard WB-1 bound (that we refer to as the *weak* WB-1 bound) only considers a single such partitioning tree. It is well-known (see, e.g., [12, 28, 18]), that the gap between $\mathsf{OPT}(X)$ and the weak WB-1 bound for an access sequence X may be as large as $\Omega(\log\log n)$. However, the "bad" access sequence X used to obtain this gap is highly dependent on the fixed partitioning tree T. It is then natural to consider a stronger variant of WB-1, that we refer to as *strong* WB-1 bound and denote by $\mathsf{WB}(X)$, that maximizes the weak WB-1 bound over all such partitioning trees. As suggested by Iacono [18], and by Kozma [20], this gives a promising approach for improving the $O(\log\log n)$-approximation factor.

1.1.3 Our results

In this paper, we show that, even for this strong variant of WB-1, the gap between $\mathsf{OPT}(X)$ and $\mathsf{WB}(X)$ may be as large as $\Omega(\log\log n/\log\log\log n)$. This negative result extends to an even stronger variant of WB-1 that we discuss below.

Our second set of results is algorithmic. We show, for any positive integer D, an (offline) D-approximation algorithm that runs in time $\mathrm{poly}(m)\cdot\exp\left(O(n^{1/2^{\Omega(D)}}\log n)\right)$. When D is constant, we obtain an $O(1)$-approximation in subexponential time. When D is $\Theta(\log\log n)$, our algorithm matches current polynomial-time approximation ratio, which is $O(\log\log n)$. In the latter case, we can also adapt the algorithm to the online setting, obtaining an $O(\log\log n)$-competitive online algorithm.

All our results use the geometric interpretation of the problem, introduced by Demaine et al. [11], leading to clean divide-and-conquer-style arguments that avoid, for example, the notion of pointers and rotations. We feel that this approach, in addition to providing a cleaner and simpler view of the problem, is more natural to work with in the context of approximation algorithms, and should be more amenable to the powerful geometric techniques in the field.

1.2 Independent work

Independently from our work, Lecomte and Weinstein [21] showed that second Wilber bound (also called *funnel bound*) dominates WB-1, and moreover, they show an access sequence X for which the two bounds have a gap of $\Omega(\log\log n)$. In particular, their result implies that the gap between $\mathsf{WB}(X)$ and $\mathsf{OPT}(X)$ is $\Omega(\log\log n)$ for that access sequence. Their result subsumes our Theorem 1.1 entirely (but not the extension discussed in Section 1.3.3).

We note that the access sequence X used in our negative results provides a gap of

$$\Omega(\log\log n/\log\log\log n)$$

between the WB-2 and the WB-1 bounds, although we only realized this after hearing the statement of the results of [21]. Additionally, Lecomte and Weinstein show that WB-2 is invariant under rotations, and use this to show that, when the WB-2 is constant, then the Independent Rectangle bound of [11] is linear.

1.3 Statements of our results

We now formally state our results.

1.3.1 Geometric representation

We use the geometric interpretation of the problem, introduced by Demaine et al. [11], that we refer to as the Min-Sat problem. Let P be any set of points in the plane. We say that two points $p, q \in P$ are *aligned* iff either their x-coordinates are equal, or their y-coordinates are equal. If p and q are not aligned, then let $\square_{p,q}$ be the smallest closed axis-aligned rectangle containing both p and q; notice that p and q must be diagonally opposite corners of this rectangle. We say that the pair (p, q) of points is *satisfied* in P iff there is some additional point $r \neq p, q$ in P that lies in $\square_{p,q}$ (the point may lie on the boundary of the rectangle). Lastly, we say that the set P of points is satisfied iff for every pair $p, q \in P$ of distinct points, either p and q are aligned, or they are satisfied in P.

In the Min-Sat problem, the input is a set P of points in the plane with integral x- and y-coordinates; we assume that all x-coordinates are between 1 and n, and all y-coordinates are between 1 and m and distinct from each other, and that $|P| = m$. The goal is to find a minimum-cardinality set Y of points, such that the set $P \cup Y$ of points is satisfied.

An access sequence X over keys $\{1, \dots, n\}$ can be represented by a set P of points in the plane as follows: if a key x is accessed at time y, then add the point (x, y) to P. Demaine et al. [11] showed that, for every access sequence X, if we denote by P the corresponding set of points in the plane, then the value of the optimal solution to the Min-Sat problem on P is $\Theta(\mathsf{OPT}(X))$. They also showed that, in order to obtain an $O(\alpha)$-approximation algorithm for BST's, it is sufficient to obtain an α-approximation algorithm for the Min-Sat problem. In the online version of the Min-Sat problem, at every time step t, we discover the unique input point whose y-coordinate is t, and we need to make an irrevocable decision on which points with y-coordinate t to add to the solution. Demaine et al. [11] also showed that an α-competitive online algorithm for Min-Sat implies an $O(\alpha)$-competitive online algorithm for BST's. For convenience, we do not distinguish between the input access sequence X and the corresponding set of points in the plane, that we also denote by X.

1.3.2 Negative results for WB-1

We say that an input access sequence X is a *permutation* if each key in $\{1, \ldots, n\}$ is accessed exactly once. Equivalently, in the geometric view, every column with an integral x-coordinate contains exactly one input point.

Informally, the WB-1 bound for an input sequence X is defined as follows. Let B be the bounding box containing all points of X, and consider any vertical line L drawn across B, that partitions it into two vertical strips, separating the points of X into two subsets X_1 and X_2. Assume that the points of X are ordered by their y-coordinates from smallest to largest. We say that a pair $(x, x') \in X$ of points *cross* the line L, iff x and x' are consecutive points of X, and they lie on different sides of L. Let $C(L)$ be the number of all pairs of points in X that cross L. We then continue this process recursively with X_1 and X_2, with the final value of the WB-1 bound being the sum of the two resulting bounds obtained for X_1 and X_2, and $C(L)$. This recursive partitioning process can be represented by a binary tree T that we call a *partitioning tree* (we note that the partitioning tree is not related to the BST tree that the BST algorithm maintains). Every vertex v of the partitioning tree is associated with a vertical strip $S(v)$, where for the root vertex r, $S(r) = B$. If the partitioning algorithm uses a vertical line L to partition the strip $S(v)$ into two sub-strips S_1 and S_2, then vertex v has two children, whose corresponding strips are S_1 and S_2. Note that every sequence of vertical lines used in the recursive partitioning procedure corresponds to a unique partitioning tree and vice versa. Given a set X of points and a partitioning tree T, we denote by $\mathsf{WB}_T(X)$ the WB-1 bound obtained for X while following the partitioning scheme defined by T. Wilber [29] showed that, for every partitioning tree T, $\mathsf{OPT}(X) \geq \Omega(\mathsf{WB}_T(X))$ holds. Moreover, Demaine et al. [12] showed that, if T is a balanced tree, then $\mathsf{OPT}(X) \leq O(\log\log n) \cdot \mathsf{WB}_T(X)$. These two bounds are used to obtain the $O(\log\log n)$-competitive algorithm of [12]. We call this variant of WB-1, that is defined with respect to a fixed tree T, the *weak* WB-1 bound.

Unfortunately, it is well-known (see, e.g., [12, 28, 18]), that the gap between $\mathsf{OPT}(X)$ and the weak WB-1 bound on an input X may be as large as $\Omega(\log\log n)$. In other words, for any fixed partitioning tree T, there exists an input X (that depends on T), with $\mathsf{WB}_T(X) \leq O(\mathsf{OPT}(X)/\log\log n)$. However, the construction of this "bad" input X depends on the fixed partitioning tree T.

We consider a stronger variant of WB-1, that we refer to as *strong* WB-1 bound and denote by $\mathsf{WB}(X)$, that maximizes the weak WB-1 bound over all such partitioning trees, that is, $\mathsf{WB}(X) = \max_T\{\mathsf{WB}_T(X)\}$.

Using this stronger bound as an alternative to weak WB-1 in order to obtain better approximation algorithms was suggested by Iacono [18], and by Kozma [20].

Our first result rules out this approach: we show that, even for the strong WB-1 bound, the gap between $\mathsf{WB}(X)$ and $\mathsf{OPT}(X)$ may be as large as $\Omega(\log\log n/\log\log\log n)$, even if the input X is a permutation.

Theorem 1.1. *For infinitely many integer n, there exists an access sequence X on n keys with $|X| = n$,* *such that X is a permutation,* $\mathsf{OPT}(X) \geq \Omega(n \log\log n)$*, but* $\mathsf{WB}(X) \leq O(n \log\log\log n)$.

In particular, for every partitioning tree T, $\frac{\mathsf{OPT}(X)}{\mathsf{WB}_T(X)} \geq \Omega\left(\frac{\log\log n}{\log\log\log n}\right)$ for infinitely many sequences X. We note that it is well known (see, e.g., [6]), that any c-approximation algorithm for permutation input can be turned into an $O(c)$-approximation algorithm for any input sequence. However, the known instances that achieve an $\Omega(\log\log n)$-gap between the weak WB-1 bound and OPT are not permutations. Therefore, our result is the first to provide a super-constant gap between WB-1 and OPT for permutations, even for the case of weak WB-1.

1.3.3 Extension of WB-1

We consider natural generalizations of the WB-1 bound that allow partitioning the plane both horizontally and vertically. We call the new bounds the *consistent Guillotine Bound* and the *Guillotine Bound*. Our negative result extends to the consistent Guillotine Bound but *not* to the Guillotine Bound. The Guillotine Bound seems to maintain the algorithm-friendly nature of WB-1, and in particular it naturally fits into the algorithmic framework that we propose. We hope that this bound can lead to improved algorithms, both in the offline and the online settings

1.3.4 Separating the two Wilber bounds

The sequence X given by Theorem 1.1 not only provides a separation between WB-1 and OPT, but it also provides a separation between WB-1 and WB-2. The latter can be defined in the geometric view as follows. Recall that, for a pair of points $x, y \in X$, $\square_{x,y}$ is the smallest closed rectangle containing both x and y. For a point x in the access sequence X, the *funnel* of x is the set of all points $y \in X$, for which $\square_{x,y}$ does not contain any point of $X \setminus \{x, y\}$, and $\text{alt}(x)$ is the number of alterations between the left of x and the right of x in the funnel of x. The second Wilber Bound for sequence X is then defined as: $\mathsf{WB}^{(2)}(X) = |X| + \sum_{x \in X} \text{alt}(x)$. We show that, for the sequence X given by Theorem 1.1, $\mathsf{WB}^{(2)}(X) \geq \Omega(n \log\log n)$ holds, and therefore $\mathsf{WB}^{(2)}(X)/\mathsf{WB}(X) \geq \Omega(\log\log n/\log\log\log n)$ for that sequence, implying that the gap between $\mathsf{WB}(X)$ and $\mathsf{WB}^{(2)}(X)$ may be as large as $\Omega(\log\log n/\log\log\log n)$. We note that we only realized that our results provide this stronger separation between the two Wilber bounds after hearing the statements of the results from the independent work of Lecomte and Weinstein [21] mentioned above.

1.3.5 Algorithmic results

We provide new simple approximation algorithms for the problem, that rely on its geometric interpretation, namely the Min-Sat problem.

Theorem 1.2. *There is an offline algorithm for* Min-Sat*, that, given any integer $D \geq 1$, and an access sequence X of length m to n keys, produces a solution of cost at most $D \cdot \mathsf{OPT}(X)$ and has running time*

$\mathrm{poly}(m) \cdot \exp\left(O\left(n^{1/2^{\Omega(D)}} \log n\right)\right)$. *For* $D = \Theta(\log\log n)$, *the algorithm's running time is polynomial in* n *and* m, *and it can be adapted to the online setting, achieving an* $O(\log\log n)$*-competitive ratio.*

While the $O(\log\log n)$-approximation factor achieved by our algorithm in time $\mathrm{poly}(mn)$ is similar to that achieved by other known algorithms [12, 16, 28], this is the first algorithm that relies solely on the geometric formulation of the problem, which is arguably cleaner, simpler, and better suited for exploiting the rich toolkit of algorithmic techniques developed in the areas of online and approximation algorithms.

1.3.6 Erratum

Some erroneous complexity theoretic inferences, made in the paragraph following the statement of Theorem 2 (page 33:6) in the conference version of this paper [4], are retracted in the present article. In particular, at this time we are unable to rule out, under any plausible complexity-theoretic assumption, the possibility that constant-factor approximation of Min-Sat is NP-hard.

The main results are not affected and are identical in the two versions.

2 Preliminaries

All our results only use the geometric interpretation of the problem, that we refer to as the Min-Sat problem. We include the formal definition of algorithms for BST's and formally state their equivalence.

2.1 The Min-Sat problem

For a point $p \in \mathbb{R}^2$ in the plane, we denote by $p.x$ and $p.y$ its x- and y-coordinates, respectively. Given any pair p, p' of points, we say that they are *aligned* if $p.x = p'.x$ or $p.y = p'.y$. If p and p' are not aligned, then we let $\square_{p,p'}$ be the smallest closed axis-aligned rectangle containing both p and p'; note that p and p' must be diagonally opposite corners of the rectangle.

Definition 2.1. We say that a non-aligned pair p, p' of points is *satisfied by a point* p'' if p'' is distinct from p and p' and $p'' \in \square_{p,p'}$ (where p'' may lie on the boundary of the rectangle). We say that a set S of points is *satisfied* if for every non-aligned pair $p, p' \in S$ of points, there is some point $p'' \in S$ that satisfies this pair.

We refer to horizontal and vertical lines as *rows* and *columns* respectively. For a collection of points X, the *active rows* of X are the rows that contain at least one point in X. We define the notion of *active columns* analogously. We denote by $r(X)$ and $c(X)$ the number of active rows and active columns of the point set X, respectively. We say that a point set X is a *semi-permutation* if every active row contains exactly one point of X. Note that, if X is a semi-permutation, then $c(X) \leq r(X)$. We say that X is a *permutation* if it is a semi-permutation, and additionally, every active column contains exactly one point of X. Clearly, if X is a permutation, then $c(X) = r(X) = |X|$. We denote by B the smallest closed rectangle containing all points of X, and call B the *bounding box*.

We are now ready to define the Min-Sat problem. The input to the problem is a set X of points that is a semi-permutation, and the goal is to compute a minimum-cardinality set Y of points, such that $X \cup Y$ is satisfied. We say that a set Y of points is a *feasible solution* for X if $X \cup Y$ is satisfied. We denote by $\mathsf{OPT}(X)$ the minimum value $|Y|$ of any feasible solution Y for X.[2]

In the online version of the Min-Sat problem, at every time step t, we discover the unique input point from X whose y-coordinate is t, and we need to make an irrevocable decision on which points with y-coordinate t to add to the solution Y. As shown by Demaine et al. [11], the Min-Sat problem is equivalent to the BST problem, in the following sense:

Theorem 2.2 (Demaine et al.)**.** *Any efficient α-approximation algorithm for* Min-Sat *can be transformed into an efficient $O(\alpha)$-approximation algorithm for BST's, and similarly any online α-competitive algorithm for* Min-Sat *can be transformed into an online $O(\alpha)$-competitive algorithm for BST's.*

2.2 Basic geometric properties

The following observation is well known (see, e.g., Observation 2.1 from [11]).

Observation 2.3. Let Z be any satisfied point set. Then for every pair $p, q \in Z$ of distinct points, there is a point $r \in \square_{p,q} \setminus \{p, q\}$ such that $r.x = p.x$ or $r.y = p.y$.

Proof. Since the set Z is satisfied, rectangle $\square_{p,q}$ must contain at least one point of Z that is distinct from p and q. Among all such points, let r be the one with smallest ℓ_1-distance to p. We claim that either $p.x = r.x$, or $p.y = r.y$. Indeed, assume otherwise. Then p and r are not aligned, but no point of Z lies in $\square_{p,r} \setminus \{p, r\}$, contradicting the fact that Z is a satisfied point set. □

2.2.1 Collapsing sets of columns or rows

Assume that we are given any set X of points, and any collection $\mathcal{C}$ of consecutive active columns for X. In order to collapse the set $\mathcal{C}$ of columns, we replace $\mathcal{C}$ with a single representative column C (for concreteness, we use the column of $\mathcal{C}$ with minimum x-coordinate). For every point $p \in X$ that lies on a column of $\mathcal{C}$, we replace p with a new point, lying on the column C, whose y-coordinate remains the same. Formally, we replace point p with point $(x, p.y)$, where

[2]In the original paper that introduced this problem [11], the value of the solution is defined as $|X \cup Y|$, while our solution value is $|Y|$. For the purpose of showing the results in this paper, the two definitions are equivalent to within a factor of 2.

x is the x-coordinate of the column C. We denote by $X_{|\mathcal{C}}$ the resulting new set of points. We can similarly define collapsing set of rows. The following useful observation is easy to verify.

Observation 2.4. Let S be any set of points, and let $\mathcal{C}$ be any collection of consecutive active columns (or rows) with respect to S. If S is a satisfied set of points, then so is $S_{|\mathcal{C}}$.

Proof. It is sufficient to prove the observation for the case where $\mathcal{C}$ contains two consecutive active columns, that we denote by C and C', which are collapsed into the column C. We can then apply this argument iteratively to collapse any number of columns.

Assume for contradiction that the set $S_{|\mathcal{C}}$ of points is not satisfied, and let $p, q \in S_{|\mathcal{C}}$ be a pair of points that are not satisfied. Note that, if p and q cannot both lie on the column C in $S_{|\mathcal{C}}$. Moreover, if both p and q lie to the right, or to the left of the column C, then they continue to be satisfied by the same point $r \in S$ that satisfied them in set S. We now consider two cases.

Assume first that p lies to the left of the column C, and q lies to the right of the column C in point set $S_{|\mathcal{C}}$. Let r be the point that satisfied the pair (p, q) in point set S. If r lied on column C in S, then it remains on column C in $S_{|\mathcal{C}}$. If r lied on column C' in S, then a copy of r lies on column C in $S_{|\mathcal{C}}$, and this copy continues to satisfy the pair (p, q). Otherwise, point r belongs to set $S_{|\mathcal{C}}$, and it continues to satisfy the pair (p, q).

It now remains to consider the case when exactly one of the two points (say p) lies on the column C in $S_{|\mathcal{C}}$. Assume w.l.o.g. that q lies to the right of p and below it in $S_{|\mathcal{C}}$. Then either p belongs to S (in which case we denote $p' = p$), or p is a copy of some point p' that lies on column C' in S. Let r be the point that satisfies the pair (p', q) of points in S. Using the same reasoning as before, it is easy to see that either r belongs to $S_{|\mathcal{C}}$, where it continues to satisfy the pair (p, q) of points, or a copy of r belongs to $S_{|\mathcal{C}}$, and it also continues to satisfy the pair (p, q).

It is easy to verify that an analogue of Observation 2.4 holds for collapsing rows as well. □

2.2.2 Canonical solutions

We say that a solution Y for input X is *canonical* iff every point $p \in Y$ lies on an active row and an active column of X. It is easy to see that *any* solution can be transformed into a canonical solution, without increasing its cost.

Observation 2.5. There is an efficient algorithm, that, given an instance X of Min-Sat and any feasible solution Y for X, computes a feasible canonical solution $\hat{Y}$ for X with $|\hat{Y}| \leq |Y|$.

Proof. Let C and C' be any pair of consecutive active columns for X, such that some point of Y lies strictly between C and C'. Let $\mathcal{C}$ be the set of all columns that lie between C and C', including C but excluding C', that contain points of $X \cup Y$. We collapse the columns in $\mathcal{C}$ into the column C, obtaining a new feasible solution for instance X (we use Observation 2.4). We continue this process until every point of the resulting solution Y lies on an active column, and we perform the same procedure for the rows. □

2.3 Partitioning trees

We now turn to define partitioning trees, that are central to both defining the WB-1 bound and to describing our algorithm.

Let X be the a set of points that is a semi-permutation. We can assume without loss of generality that every column with an integral x-coordinate between 1 and $c(X)$ inclusive contains at least one point of X. Let B be the bounding box of X. Assume that the set of active columns is $\{C_1, \ldots, C_a\}$, where $a = c(X)$, and that for all $1 \le i \le a$, the x-coordinate of column C_i is i. Let $\mathcal{L}$ be the set of all vertical lines with half-integral x-coordinates between $1 + 1/2$ and $a - 1/2$ (inclusive). Throughout, we refer to the vertical lines in $\mathcal{L}$ as *auxiliary columns*. Let σ be an arbitrary ordering of the lines of $\mathcal{L}$ and denote $\sigma = (L_1, L_2, \ldots, L_{a-1})$. We define a hierarchical partition of the bounding box B into vertical strips using σ, as follows. We perform $a - 1$ iterations. In the first iteration, we partition the bounding box B, using the line L_1, into two vertical strips, S_L and S_B. For $1 < i \le a - 1$, in iteration i we consider the line L_i, and we let S be the unique vertical strip in the current partition that contains the line L_i. We then partition S into two vertical sub-strips by the line L_i. When the partitioning algorithm terminates, every vertical strip contains exactly one active column.

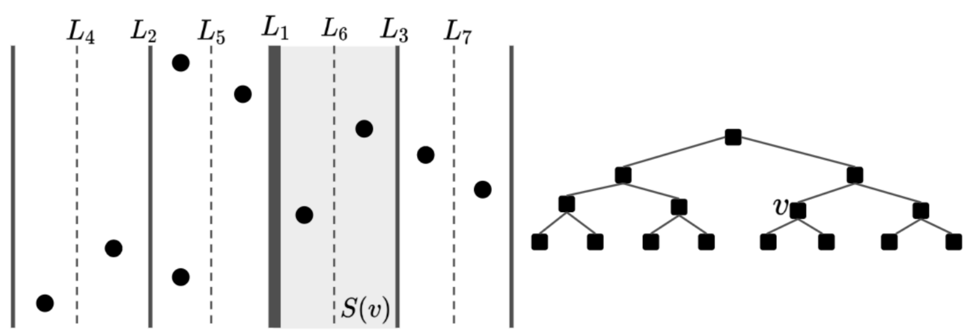


Figure 1: An Illustration of a partitioning tree and the corresponding sequence $\sigma = (L_1, \ldots, L_7)$. Strip $S(v)$ corresponds to node v that owns line L_6.

This partitioning process can be naturally described by a binary tree $T = T(\sigma)$, that we call a *partitioning tree* associated with the ordering σ (see Figure 1). Each node $v \in V(T)$ is associated with a vertical strip $S(v)$ of the bounding box B. The strip $S(r)$ of the root vertex r of T is the bounding box B. For every inner vertex $v \in V(T)$, if $S = S(v)$ is the vertical strip associated with v, and if $L \in \mathcal{L}$ is the first line in σ that lies strictly in S, then line L partitions S into two sub-strips, $S(v_1)$ and $S(v_2)$, corresponding to the two children v_1 and v_2 of v in the partitioning tree. We say that v *owns* the line L, and we denote $L = L(v)$. For each leaf node v, the corresponding strip $S(v)$ contains exactly one active column of X, and v does not own any line of $\mathcal{L}$. For each vertex $v \in V(T)$, let $N(v) = |X \cap S(v)|$ be the number of points from X that lie in $S(v)$, and let width(v) be the width of the strip $S(v)$. Given a partition tree T for point set X, we refer to the vertical strips in $\{S(v)\}_{v \in T}$ as T-strips.

2.4 The WB-1 bound

The WB-1 bound[3] is defined with respect to an ordering (or a permutation) σ of the auxiliary columns, or, equivalently, with respect to the partitioning tree $T(\sigma)$. It will be helpful to keep both these views in mind. In this paper, we will make a clear distinction between a weak variant of the WB-1 bound, as defined by Wilber himself in [29] and a strong variant, as mentioned in [18].

Let X be a semi-permutation, and let $\mathcal{L}$ be the corresponding set of auxiliary columns. Consider an arbitrary fixed ordering σ of columns in $\mathcal{L}$ and its corresponding partition tree $T = T(\sigma)$. For each inner node $v \in V(T)$, consider the set $X' = X \cap S(v)$ of input points that lie in the strip $S(v)$, and let $L(v) \in \mathcal{L}$ be the line that v owns. We denote $X' = \{p_1, p_2, \ldots, p_k\}$, where the points are indexed in the increasing order of their y-coordinates; since X is a semi-permutation, no two points of X may have the same y-coordinate. For $1 \le j < k$, we say that the ordered pair (p_j, p_{j+1}) of points form a *crossing* of $L(v)$ iff p_j, p_{j+1} lie on the opposite sides of the line $L(v)$. We let $\mathsf{cost}(v)$ be the total number of crossings of $L(v)$ by the points of $X \cap S(v)$. When $L = L(v)$, we also write $\mathsf{cost}(L)$ to denote $\mathsf{cost}(v)$. If v is a leaf vertex, then its cost is set to 0.

Definition 2.6 (WB-1 bound)**.** For any semi-permutation X, an ordering σ of the auxiliary columns in $\mathcal{L}$, and the corresponding partitioning tree $T = T(\sigma)$, the (weak) WB-1 bound of X with respect to σ is: $\mathsf{WB}_\sigma(X) = \mathsf{WB}_T(X) = \sum_{v \in V(T)} \mathsf{cost}(v)$. The strong WB-1 bound of X is $\mathsf{WB}(X) = \max_\sigma \mathsf{WB}_\sigma(X)$, where the maximum is taken over all permutations σ of the lines in $\mathcal{L}$.

It is well known that the WB-1 bound is a lower bound on the optimal solution cost:

Claim 2.7. *For any semi-permutation X, $\mathsf{WB}(X) \le 2 \cdot \mathsf{OPT}(X)$.*

The original proof of this fact is due to Wilber [29], which was later presented in the geometric view by Demaine et al. [11], via the notion of *independent rectangles*. In Section 8, we include a direct geometric proof of this fact.

We note a simple observation, that the cost can be bounded by the number of points on the smaller side.

Observation 2.8. Let X be a semi-permutation, σ an ordering of the auxiliary columns in $\mathcal{L}$, and let $T = T(\sigma)$ be the corresponding partitioning tree. Let $v \in V(T)$ be any inner vertex of the tree, whose two child vertices are denoted by v_1 and v_2. Then $\mathsf{cost}(v) \le 2\min\{|X \cap S(v_1)|, |X \cap S(v_2)|\}$.

Proof. For simplicity, we denote $X' = X \cap S(v_1)$ and $X'' = X \cap S(v_2)$. Assume w.l.o.g. that $|X'| \le |X''|$. Notice that, if the pair (p_i, p_{i+1}) of points in $S(v)$ define a crossing of $L(v)$, then one of p_i, p_{i+1} must lie in X'. Every point $p_j \in X'$ may participate in at most two pairs of points that define crossings: the pairs (p_{j-1}, p_j) and (p_j, p_{j+1}). Therefore, the total number of crossings of $L(v)$ is at most $2|X'|$. □

[3] Also called Interleaving bound [12], the first Wilber bound, "interleave lower bound" [29], or alternation bound [18]

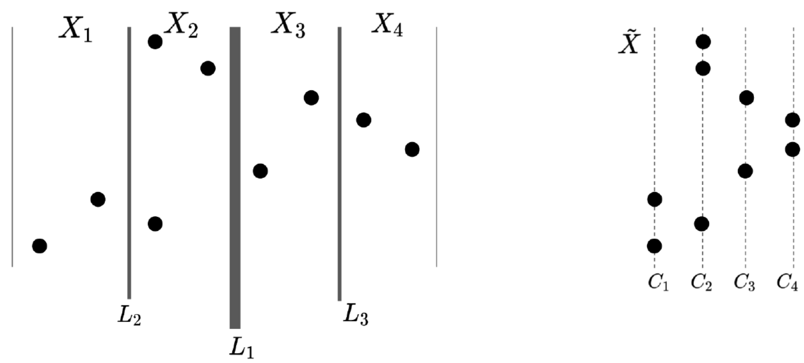


Figure 2: An illustration of a split of X by $\mathcal{L}' = \{L_1, L_2, L_3\}$.

3 Geometric decomposition theorems

In this section, we develop several technical tools that will allow us to decompose a given instance into a number of sub-instances. We then analyze the optimal solution costs and the Wilber bound values for the resulting subinstances.

3.1 Split instances

Consider a semi-permutation X. We define the **split instances** with respect to any subset $\mathcal{L}' \subseteq \mathcal{L}$ of the auxiliary columns for X: Notice that the lines in $\mathcal{L}'$ partition the bounding box B into a collection of internally disjoint strips, that we denote by $\{S_1, \ldots, S_k\}$ where $k = |\mathcal{L}'| + 1$. We can then define the **strip instances** $X_i \subseteq X$ as containing all vertices of $X \cap S_i$ for all $1 \le i \le k$, and the **compressed instance** $\tilde{X}$, that is obtained by collapsing, for each $1 \le i \le k$, all active columns that lie in strip S_i, into a single column. We call these resulting instances $(\tilde{X}, \{X_i\}_{i=1}^k)$ *a split of* X *by* $\mathcal{L}'$. See Figure 2.

Observation 3.1. Let $\mathcal{L}' \subseteq \mathcal{L}$ be a collection of lines and $(\tilde{X}, \{X_i\}_{i=1}^k)$ be a split of X by $\mathcal{L}'$. Then

- $\sum_i r(X_i) = r(X)$
- $\sum_i c(X_i)) = c(X)$
- $c(\tilde{X}) \le k$

The first property holds since X is a semi-permutation.

Consider an arbitrary ordering σ of the lines in $\mathcal{L}$, such that the lines of $\mathcal{L}'$ appear at the beginning of σ. The lines in $\mathcal{L}'$ split σ naturally into $k+1$ orderings. Let $S_1, \ldots, S_k$ be the strips obtained from partitioning box B by $\mathcal{L}'$, and for each $i \in [k]$, $\mathcal{L}_i$ is a collection of lines in strip S_i. Now, σ_i can be defined by naturally inducing σ to the lines in $\mathcal{L}_i$, and $\tilde{\sigma}$ is the ordering of lines in $\mathcal{L}'$ induced by σ. We say that $\tilde{\sigma}, \sigma_1, \dots, \sigma_k$ are the **split orderings** of σ by $\mathcal{L}'$. Similarly, the lines $\mathcal{L}'$ also split the partitioning tree $T = T(\sigma)$ into $\widetilde{T} = T(\tilde{\sigma})$ and $T_1, \dots, T_k$ where $T_i = T(\sigma_i)$ for all $i \in [k]$. These partitioning trees are called the **split partitioning trees** of T by $\mathcal{L}'$.

Observation 3.2. For $\mathcal{L}' \subseteq \mathcal{L}$ and ordering σ, let $(\tilde{X}, \{X_i\}_{i=1}^k)$ and $(\tilde{\sigma}, \{\sigma_i\})$ be the split instances and split orderings respectively. Then,

$$\sum_{i=1}^{k} \mathsf{WB}_{\sigma_i}(X_i) + \mathsf{WB}_{\tilde{\sigma}}(\tilde{X}) = \mathsf{WB}_{\sigma}(X)$$

This property can be viewed as a "perfect" decomposition property of the weak WB-1 bound with respect to the split operation.

3.2 Decomposition theorem for the optimal solution

We prove the following recurrence about the "subadditivity" property of OPT under the decomposition into split instances.

Theorem 3.3. *Let $\mathcal{L}' \subseteq \mathcal{L}$ be a collection of lines and $(\tilde{X}, \{X_i\}_{i=1}^k)$ be a split instance of X by $\mathcal{L}'$. Then*

$$\sum_{i=1}^{k} \mathsf{OPT}(X_i) + \mathsf{OPT}(\tilde{X}) \leq \mathsf{OPT}(X).$$

Proof. Let $\{S_i\}_{i=1}^k$ be the strips partitioned by $\mathcal{L}'$. Let Y be an optimal canonical solution for X, so that every point of Y lies on an active row and an active column for X. For each i, let Y_i denote the set of points of Y that lie in the strip S_i. Since these points lie in the interior of the strip, $Y = \bigcup_{i=1}^k Y_i$.

For each i, let $\mathcal{R}_i$ denote the set of all rows R, such that: (i) R contains a point of X; (ii) R contains no point of X_i; and (iii) at least one point of Y_i lies on R. Let $m_i = |\mathcal{R}_i|$. We need the following claim.

Claim 3.4. *There is a feasible solution $\hat{Y}$ to instance $\tilde{X}$, containing at most $\sum_i m_i$ points.*

Proof. We construct the solution $\hat{Y}$ for $\tilde{X}$ as follows. Consider $i \in [k]$. Let C_i be the unique column into which the columns lying in the strip S_i were collapsed. For every point $p \in Y_i$ that lies on a row $R \in \mathcal{R}_i$, we add a new point $\varphi(p)$ on the intersection of row R and column C_i to the solution $\hat{Y}$. Once we process all strips $i \in [k]$, we obtain a final set of points $\hat{Y}$. It is easy to verify that $|\hat{Y}| = \sum_i m_i$. In order to see that $\hat{Y}$ is a feasible solution to instance $\tilde{X}$, it is enough to show that the set $\tilde{X} \cup \hat{Y}$ of points is satisfied. Notice that set $X \cup Y$ of points is satisfied, and set $\tilde{X} \cup \hat{Y}$ is obtained from $X \cup Y$ by collapsing sets of active columns lying in each strip S_i for $i \in [k]$. From Observation 2.4, the point set $\tilde{X} \cup \hat{Y}$ is satisfied. ◻

We now consider the strip instances $\{X_i\}_{i\in[k]}$ and prove the following claim, that will complete the proof of the lemma.

Claim 3.5. *For each $i \in [k]$, $\mathsf{OPT}(X_i) \le |Y_i| - m_i$.*

Proof. Notice first that the point set $X_i \cup Y_i$ must be satisfied. We will modify point set Y_i, to obtain another set Y'_i, so that Y'_i remains a feasible solution for X_i, and $|Y'_i| \le |Y_i| - m_i$.

In order to do so, we perform m_i iterations. In each iteration, we will decrease the size of Y_i by at least one, while also decreasing the cardinality of the set $\mathcal{R}_i$ of rows by exactly 1, and maintaining the feasibility of the solution Y_i for X_i.

In every iteration, we select two arbitrary rows R and R', such that: (i) $R \in \mathcal{R}_i$; (ii) R' is an active row for instance X_i, and (iii) no point of $Y_i \cup X_i$ lies strictly between rows R and R'. We collapse the rows R and R' into the row R'. From Observation 2.4, the resulting new set Y_i of points remains a feasible solution for instance X_i. We claim that $|Y_i|$ decreases by at least 1. In order to show this, it is enough to show that there are two points $p, p' \in X_i \cup Y_i$, with $p \in R$, $p' \in R'$, such that the x-coordinates of p and p' are the same; in this case, after we collapse the rows, x and x' are mapped to the same point. Assume for contradiction that no such two points exist. Let $p \in R \cap (X_i \cup Y_i)$, $p' \in R' \cap Y_i$ be a pair of points with smallest horizontal distance. Such points must exist since R contains a point of X_i and R' contains a point of Y_i. But then no other point of $X_i \cup Y_i$ lies in $\square_{p,p'}$, so the pair (p, p') is not satisfied in $X_i \cup Y_i$, a contradiction. $\square$

$\square$

3.3 Decomposition theorem for the strong WB-1 bound.

We prove the following recurrence about the strong WB-1 bound bound.

Theorem 3.6. *Let $\mathcal{L}' \subseteq \mathcal{L}$ be a collection of lines and $(\tilde{X}, \{X_i\}_{i=1}^k)$ be a split instance of X by $\mathcal{L}'$. Then*

$$\mathsf{WB}(X) \le 4\mathsf{WB}(\tilde{X}) + 8\sum_{i=1}^{k} \mathsf{WB}(X_i) + O(|X|).$$

We find this result somewhat surprising. One can think of the expression $\mathsf{WB}(\widetilde{X}) + \sum_{i\in[k]} \mathsf{WB}(X_i)$ as a WB-1 bound obtained by first cutting along the lines that serve as boundaries of the strips S_i for $i \in [k]$, and then starting to cut inside the individual strips afterwards. However, $\mathsf{WB}(X)$ is the maximum of $\mathsf{WB}_\sigma(X)$ obtained over all possible orderings σ, including those that do not obey this cutting order.

The remainder of this section is dedicated to the proof of Theorem 3.6. For each $1 \le i \le k$, we denote by $\mathcal{B}_i$ be the set of consecutive active columns containing the points of X_i, and we refer to it as a *block*. For brevity, we also say "Wilber bound" to mean the strong WB-1 bound in this section.

3.3.1 Forbidden points

For the sake of the proof, we need the notion of forbidden points. Let $\hat{X}$ be some semi-permutation and $\hat{\mathcal{L}}$ be the set of auxiliary columns for $\hat{X}$. Let $F \subseteq \hat{X}$ be a set of points that we refer to as *forbidden points*. We now define the strong WB-1 bound with respect to the forbidden points, $\mathsf{WB}^F(\hat{X})$.

Consider any permutation $\hat{\sigma}$ of the lines in $\hat{\mathcal{L}}$. Intuitively, $\mathsf{WB}^F_{\hat{\sigma}}(\hat{X})$ counts all the crossings contributed to $\mathsf{WB}_{\hat{\sigma}}(\hat{X})$ but excludes all crossing pairs (p, p') where at least one of p, p' lie in F. Similar to $\mathsf{WB}(X)$, we define $\mathsf{WB}^F(\hat{X}) = \max_{\hat{\sigma}} \mathsf{WB}^F_{\hat{\sigma}}(\hat{X})$, where the maximum is over all permutations $\hat{\sigma}$ of the lines in $\hat{\mathcal{L}}$.

Next, we define $\mathsf{WB}^F_{\hat{\sigma}}(\hat{X})$ more formally. Let $T = T(\hat{\sigma})$ be the partitioning tree associated with $\hat{\sigma}$. For each vertex $v \in V(T)$, let $L = L(v)$ be the line that belongs to v, and let $\mathrm{Cr}_{\hat{\sigma}}(L)$ be the set of all crossings (p, p') that contribute to $\mathsf{cost}(L)$; that is, p and p' are two points that lie in the strip $S(v)$ on two opposite sides of L, and no other point of $\hat{X} \cap S(v)$ lies between the row of p and the row of p'. Let $\mathrm{Cr}_{\hat{\sigma}} = \bigcup_{L \in \hat{\mathcal{L}}} \mathrm{Cr}_{\hat{\sigma}}(L)$. Observe that $\mathsf{WB}_{\hat{\sigma}}(X) = |\mathrm{Cr}_{\hat{\sigma}}|$ by definition. We say that a crossing $(p, p') \in \mathrm{Cr}_{\hat{\sigma}}(L)$ is *forbidden* iff at least one of p, p' lie in F; otherwise the crossing is *allowed*. We let $\mathrm{Cr}^F_{\hat{\sigma}}(L)$ be the set of crossings obtained from $\mathrm{Cr}_{\hat{\sigma}}(L)$ by discarding all forbidden crossings. We then let $\mathrm{Cr}^F_{\hat{\sigma}} = \bigcup_{L \in \hat{\mathcal{L}}} \mathrm{Cr}^F_{\hat{\sigma}}(L)$, and $\mathsf{WB}^F_{\hat{\sigma}}(\hat{X}) = |\mathrm{Cr}^F_{\hat{\sigma}}|$.

We emphasize that $\mathsf{WB}^F(\hat{X})$ is not necessarily the same as $\mathsf{WB}(\hat{X} \setminus F)$, as some crossings of the instance $\hat{X} \setminus F$ may not correspond to allowed crossings of instance $\hat{X}$.

3.3.2 Proof overview and notation

Consider first the compressed instance $\tilde{X}$, that is a semi-permutation. We denote its set of active columns by $\tilde{\mathcal{C}} = \{C_1, \ldots, C_k\}$, where the columns are indexed in their natural left-to-right order. Therefore, C_i is the column that was obtained by collapsing all active columns in strip S_i. It would be convenient for us to slightly modify the instance $\tilde{X}$ by simply multiplying all x-coordinates of the points in $\tilde{X}$ and of the columns in $\tilde{\mathcal{C}}$ by factor 2. Note that this does not affect the value of the optimal solution or of the Wilber bound, but it ensures that every consecutive pair of columns in $\tilde{\mathcal{C}}$ is separated by a column with an integral x-coordinate. We let $\tilde{\mathcal{L}}$ be the set of all vertical lines with half-integral coordinates in the resulting instance $\tilde{X}$.

Similarly, we modify the original instance X, by inserting, for every consecutive pair $\mathcal{B}_i, \mathcal{B}_{i+1}$ of blocks, a new column with an integral coordinate that lies between the columns of $\mathcal{B}_i$ and the columns of $\mathcal{B}_{i+1}$. This transformation does not affect the optimal solution cost or the value of the Wilber bound. For all $1 \leq i \leq N$, we denote $q_i = |\mathcal{B}_i|$. We denote by $\mathcal{L}$ the set of all vertical lines with half-integral coordinates in the resulting instance X.

Consider any block $\mathcal{B}_i$. We denote by $\mathcal{L}_i = \left\{L_i^1, \ldots, L_i^{q_i+1}\right\}$ the set of $q_i + 1$ consecutive vertical lines in $\mathcal{L}$, where L_i^1 appears immediately before the first column of $\mathcal{B}_i$, and $L_i^{q_i+1}$ appears immediately after the last column of $\mathcal{B}_i$. Notice that $\mathcal{L} = \bigcup_{i=1}^{N} \mathcal{L}_i$.

Recall that our goal is to show that $\mathsf{WB}(X) \leq 4\mathsf{WB}(\tilde{X}) + 8\sum_{i=1}^{k} \mathsf{WB}(X_i) + O(|X|)$. In order to do so, we fix a permutation σ of $\mathcal{L}$ that maximizes $\mathsf{WB}_\sigma(X)$, so that $\mathsf{WB}(X) = \mathsf{WB}_\sigma(X)$. We then gradually transform it into a permutation $\tilde{\sigma}$ of $\tilde{\mathcal{L}}$, such that $\mathsf{WB}_{\tilde{\sigma}}(\tilde{X}) \geq \mathsf{WB}_\sigma(X)/4 - 2\sum_{i=1}^{k} \mathsf{WB}(X_i) - O(|X|)$. This will prove that $\mathsf{WB}(X) \leq 4\mathsf{WB}(\tilde{X}) + 8\sum_{i=1}^{k} \mathsf{WB}(X_i) + O(|X|)$.

In order to perform this transformation, we will process every block $\mathcal{B}_i$ one-by-one. When block $\mathcal{B}_i$ is processed, we will "consolidate" all lines of $\mathcal{L}_i$, so that they will appear almost consecutively in the permutation σ, and we will show that this process does not increase the Wilber bound by too much. The final permutation that we obtain after processing every block $\mathcal{B}_i$ can then be naturally transformed into a permutation $\tilde{\sigma}$ of $\tilde{\mathcal{L}}$, whose Wilber bound cost is similar. The main challenge is to analyze the increase in the Wilber bound in every iteration. In order to facilitate the analysis, we will work with the Wilber bound with respect to forbidden points. Specifically, we will define a set $F \subseteq X$ of forbidden points, such that $\mathsf{WB}_\sigma^F(X) \geq \mathsf{WB}_\sigma(X)/4 - \sum_{i=1}^{k} \mathsf{WB}(X_i)$. For every block $\mathcal{B}_i$, we will also define a bit $b_i \in \{0, 1\}$, that will eventually guide the way in which the lines of $\mathcal{L}_i$ are consolidated. As the algorithm progresses, we will modify the set F of forbidden points by discarding some points from it, and we will show that the increase in the Wilber bound with respect to the new set F is small relatively to the original Wilber bound with respect to the old set F. We start by defining the set F of forbidden points, and the bits b_i for the blocks $\mathcal{B}_i$. We then show how to use these bits in order to transform permutation σ of $\mathcal{L}$ into a new permutation σ' of $\mathcal{L}$, which will in turn be transformed into a permutation $\tilde{\sigma}$ of $\tilde{\mathcal{L}}$.

From now on we assume that the permutation σ of the lines in $\mathcal{L}$ is fixed.

3.3.3 Defining the set of forbidden points

Consider any block $\mathcal{B}_i$, for $1 \leq i \leq k$. We denote by $L_i^* \in \mathcal{L}_i$ the vertical line that appears first in the permutation σ among all lines of $\mathcal{L}_i$, and we denote by $L_i^{**} \in \mathcal{L}_i$ the line that appears last in σ among all lines of $\mathcal{L}_i$.

We perform k iteration. In iteration i, for $1 \leq i \leq k$, we consider the block $\mathcal{B}_i$. We let $b_i \in \{0, 1\}$ be a bit chosen uniformly at random, independently from all other random bits. If $b_i = 0$, then all points of X_i that lie to the left of L_i^* are added to the set F of forbidden points; otherwise, all points of X_i that lie to the right of L_i^* are added to the set F of forbidden points. We show that the expected number of the remaining crossings is large.

Claim 3.7. *The expectation, over the choice of the bits b_i, of $|\mathrm{Cr}_\sigma^F|$ is at least $|\mathsf{WB}(X)|/4 - \sum_{i=1}^{k} \mathsf{WB}(X_i)$.*

Proof. Consider any crossing $(p, p') \in \mathrm{Cr}_\sigma$. We consider two cases. Assume first that there is some index i, such that both p and p' belong to X_i, and they lie on opposite sides of L_i^*. In this case, (p, p') becomes a forbidden crossing with probability 1. However, the total number of all such crossings is bounded by $\mathsf{WB}(X_i)$. Indeed, if we denote by $\hat{\mathcal{L}}_i$ the set of all vertical lines with half-integral coordinates for instance X_i, then permutation σ of $\mathcal{L}$ naturally induces permutation σ_i of $\hat{\mathcal{L}}_i$. Moreover, any crossing $(p, p') \in \mathrm{Cr}_\sigma$ with $p, p' \in X_i$ must also contribute to the cost of σ_i in instance X_i. Since the cost of σ_i is bounded by $WB(X_i)$, the number of crossings $(p, p') \in \mathrm{Cr}_\sigma$ with $p, p' \in X_i$ is bounded by $WB(X_i)$.

Consider now any crossing $(p, p') \in \mathrm{Cr}_\sigma$, and assume that there is no index i, such that both p and p' belong to X_i, and they lie on opposite sides of L_i^*. Then with probability at least $1/4$, this crossing remains allowed. Therefore, the expectation of $|\mathrm{Cr}_\sigma^F|$ is at least $|\mathrm{Cr}_\sigma|/4 - \sum_{i=1}^k \mathsf{WB}(X_i) = |\mathsf{WB}_\sigma(X)|/4 - \sum_{i=1}^k \mathsf{WB}(X_i) = |\mathsf{WB}(X)|/4 - \sum_{i=1}^k \mathsf{WB}(X_i)$. ☐

From the above claim, there is a choice of the bits $b_1, \ldots, b_k$, such that, if we define the set F of forbidden points with respect to these bits as before, then $|\mathrm{Cr}_\sigma^F| \geq \mathsf{WB}(X)/4 - \sum_{i=1}^k \mathsf{WB}(X_i)$. From now on we assume that the values of the bits $b_1, \ldots, b_k$ are fixed, and that the resulting set F of forbidden points satisfies that $|\mathrm{Cr}_\sigma^F| \geq \mathsf{WB}(X)/4 - \sum_{i=1}^k \mathsf{WB}(X_i)$.

3.3.4 Transforming σ into σ'

We now show how to transform the original permutation σ of $\mathcal{L}$ into a new permutation σ' of $\mathcal{L}$, which we will later transform into a permutation $\tilde{\sigma}$ of $\tilde{\mathcal{L}}$. We perform k iterations. The input to the ith iteration is a permutation σ_i of $\mathcal{L}$ and a subset $F_i \subseteq F$ of forbidden points. The output of the iteration is a new permutation σ_{i+1} of $\mathcal{L}$, and a set $F_{i+1} \subseteq F_i$ of forbidden points. The final permutation is $\sigma' = \sigma_{k+1}$, and the final set F_{k+1} of forbidden points will be empty. The input to the first iteration is $\sigma_1 = \sigma$ and $F_1 = F$. We now fix some $1 \leq i \leq k$, and show how to execute the ith iteration. Intuitively, in the ith iteration, we consolidate the lines of $\mathcal{L}_i$. Recall that we have denoted by $L_i^*, L_i^{**} \in \mathcal{L}_i$ the first and the last lines of $\mathcal{L}_i$, respectively, in the permutation σ. We only move the lines of $\mathcal{L}_i$ in iteration i, so this ensures that, in permutation σ_i, the first line of $\mathcal{L}_i$ that appears in the permutation is L_i^*, and the last line is L_i^{**}.

We now describe the ith iteration. Recall that we are given as input a permutation σ_i of the lines of $\mathcal{L}$, and a subset $F_i \subseteq F$ of forbidden points. We consider the block $\mathcal{B}_i$ and the corresponding bit b_i.

Assume first that $b_i = 0$; recall that in this case, all points of X that lie on the columns of $\mathcal{B}_i$ to the left of L_i^* are forbidden (see Figure 3). We start by switching the locations of L_i^* and L_i^1 in the permutation σ_i (recall that L_i^1 is the leftmost line in $\mathcal{L}_i$). Therefore, L_i^1 becomes the first line of $\mathcal{L}_i$ in the resulting permutation. Next, we consider the location of line L_i^{**} in σ_i, and we place the lines $L_i^{q_i+1}, L_i^2, L_i^3, \ldots, L_i^{q_i}$ in that location, in this order. This defines the new permutation σ_{i+1}.

Assume now that $b_i = 1$; recall that in this case, all points of X that lie on the columns of $\mathcal{B}_i$ to the right of L_i^* are forbidden (see Figure 3). We start by switching the locations of L_i^* and $L_i^{q_i+1}$ in the permutation σ_i (recall that $L_i^{q_i+1}$ is the rightmost line in $\mathcal{L}_i$). Therefore, $L_i^{q_i+1}$ becomes the first line of $\mathcal{L}_i$ in the resulting permutation. Next, we consider the location of line L_i^{**} in

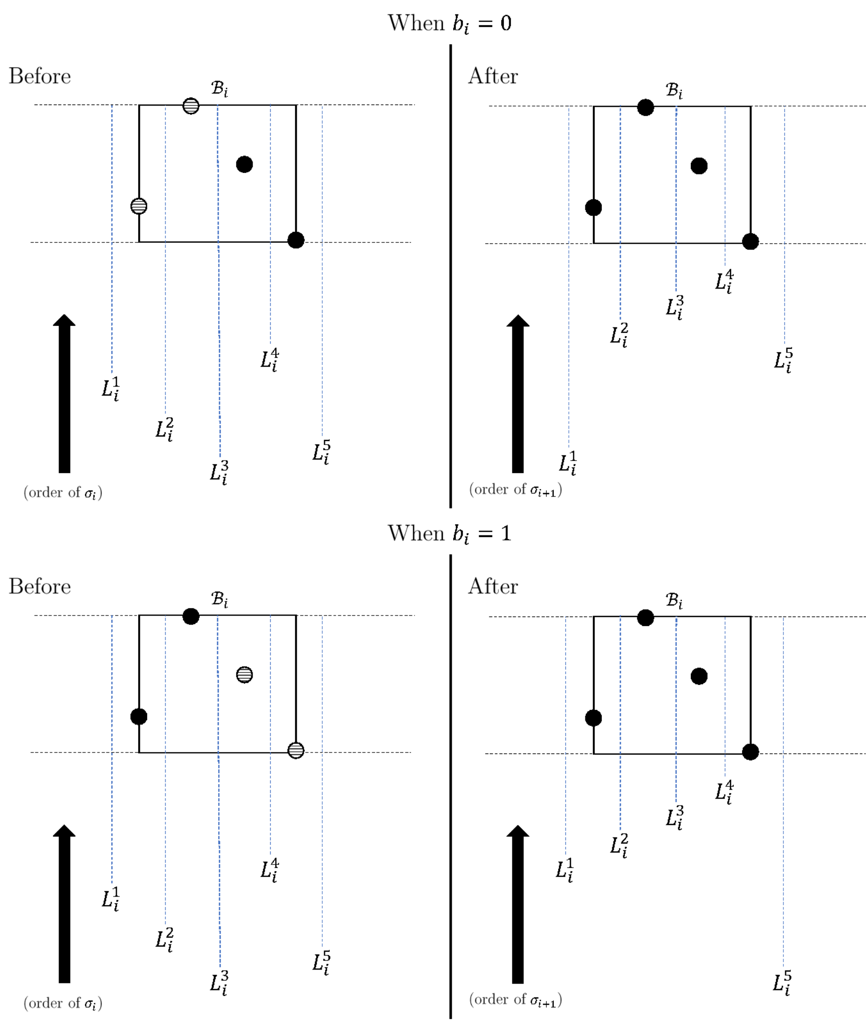


Figure 3: Modification from σ_i to σ_{i+1}. In the figure, $\mathcal{L}_i = \{L_i^1, \dots, L_i^5\}$, $L_i^* = L_i^3$ and $L_i^{**} = L_i^4$. Points with horizontal strips are forbidden.

σ_i, and we place the lines $L_i^1, L_i^2, L_i^3, \ldots, L_i^{q_i}$ in that location, in this order. This defines the new permutation σ_{i+1}.

Lastly, we discard from F_i all points that lie on the columns of $\mathcal{B}_i$, obtaining the new set F_{i+1} of forbidden points.

Once every block $\mathcal{B}_i$ is processed, we obtain a final permutation σ_{k+1} that we denote by σ', and the final set $F_{k+1} = \emptyset$ of forbidden lines. The following lemma is central to our analysis. It shows that the Wilber bound does not decrease by much after every iteration. The Wilber bound is defined with respect to the appropriate sets of forbidden points.

Lemma 3.8. *For all* $1 \le i \le k$, $\mathsf{WB}^{F_{i+1}}_{\sigma_{i+1}}(X) \ge \mathsf{WB}^{F_i}_{\sigma_i}(X) - \mathsf{WB}(X_i) - O(|X_i|)$.

Assume first that the lemma is correct. Recall that we have ensured that $\mathsf{WB}^{F_1}_{\sigma_1}(X) = \mathsf{WB}^{F}_{\sigma}(X) \ge \mathsf{WB}(X)/4 - \sum_{i=1}^{k} \mathsf{WB}(X_i)$. Since $F_{k+1} = \emptyset$, this will ensure that:

$$\mathsf{WB}_{\sigma'}(X) \ge \mathsf{WB}^{F}_{\sigma}(X) - \sum_i \mathsf{WB}(X_i) - O(|X|) \ge \mathsf{WB}(X)/4 - 2\sum_i \mathsf{WB}(X_i) - O(|X|).$$

We now focus on the proof of the lemma.

Proof. In order to simplify the notation, we denote σ_i by $\hat{\sigma}$, σ_{i+1} by $\hat{\sigma}'$. We also denote F_i by $\hat{F}$, and F_{i+1} by $\hat{F}'$.

Consider a line $L \in \mathcal{L}$. Recall that $\mathrm{Cr}_{\hat{\sigma}}(L)$ is the set of all crossings that are charged to the line L in permutation $\hat{\sigma}$. Recall that $\mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}(L) \subseteq \mathrm{Cr}_{\hat{\sigma}}(L)$ is obtained from the set $\mathrm{Cr}_{\hat{\sigma}}(L)$ of crossings, by discarding all crossings (p, p') where $p \in \hat{F}$ or $p' \in \hat{F}$ holds. The set $\mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}(L)$ of crossings is defined similarly.

We start by showing that for every line $L \in \mathcal{L}$ that does not lie in $\mathcal{L}_i$, the number of crossings charged to it does not decrease, that is, $\mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}(L) \ge \mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}(L)$.

Claim 3.9. *For every line* $L \in \mathcal{L} \setminus \mathcal{L}_i$, $\mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}(L) \ge \mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}(L)$.

Proof. Consider any line $L \in \mathcal{L} \setminus \mathcal{L}_i$. Let $v \in V(T(\hat{\sigma}))$ be the vertex of the partitioning tree $T(\hat{\sigma})$ corresponding to $\hat{\sigma}$ to which L belongs, and let $S = S(v)$ be the corresponding strip. Similarly, we define $v' \in V(T(\hat{\sigma}'))$ and $S' = S(v')$ with respect to $\hat{\sigma}'$. Recall that L_i^* is the first line of $\mathcal{L}_i$ to appear in the permutation σ, and L_i^{**} is the last such line. We now consider five cases.

- **Case 1.** The first case happens if L appears before line L_i^* in the permutation $\hat{\sigma}$. Notice that the prefixes of the permutations $\hat{\sigma}$ and $\hat{\sigma}'$ are identical up to the location in which L_i^* appears in $\hat{\sigma}$. Therefore, $S = S'$, and $\mathrm{Cr}_{\hat{\sigma}}(L) = \mathrm{Cr}_{\hat{\sigma}'}(L)$. Since $\hat{F}' \subseteq \hat{F}$, every crossing that is forbidden in $\hat{\sigma}'$ was also forbidden in $\hat{\sigma}$. So $\mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}(L) \subseteq \mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}(L)$, and $\mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}(L) \ge \mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}(L)$.

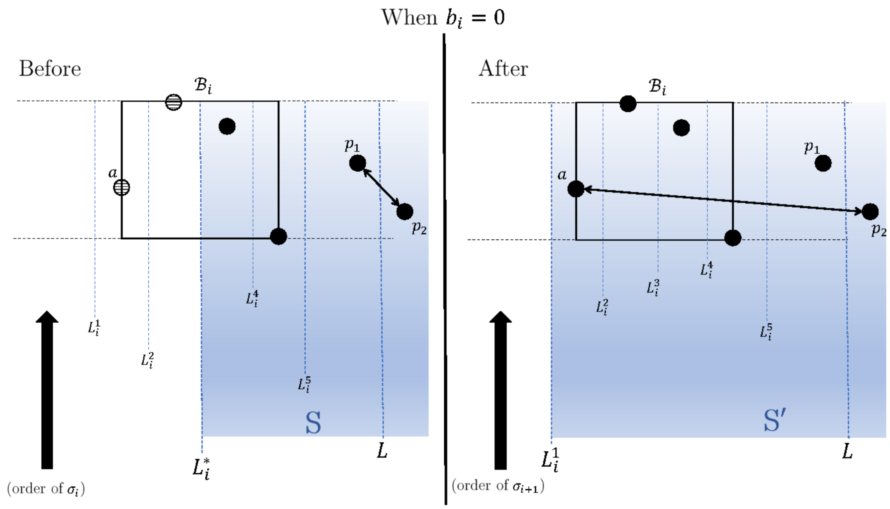


Figure 4: Illustration of the injective mapping of each crossing in $\mathrm{Cr}_{\hat{\sigma}}^{\hat{F}}(L)$ to a crossing in $\mathrm{Cr}_{\hat{\sigma}'}^{\hat{F}'}(L)$. Points with horizontal strips are forbidden points from $\hat{F}$.

- **Case 2.** The second case happens if L appears after L_i^{**} in $\hat{\sigma}$. Notice that, if we denote by $\mathcal{L}' \subseteq \mathcal{L}$ the set of all lines of $\mathcal{L}$ that lie before L in $\hat{\sigma}$, and define $\mathcal{L}''$ similarly for $\hat{\sigma}'$, then $\mathcal{L}' = \mathcal{L}''$. Therefore, $S = S'$ holds. Using the same reasoning as in Case 1, $\mathrm{Cr}_{\hat{\sigma}'}^{\hat{F}'}(L) \geq \mathrm{Cr}_{\hat{\sigma}}^{\hat{F}}(L)$.

- **Case 3.** The third case is when L appears between L_i^* and L_i^{**} in $\hat{\sigma}$, but neither boundary of the strip S belongs to $\mathcal{L}_i$. If we denote by $\mathcal{L}' \subseteq \mathcal{L} \setminus \mathcal{L}_i$ the set of all lines of $\mathcal{L} \setminus \mathcal{L}_i$ that lie before L in $\hat{\sigma}$, and define $\mathcal{L}'' \subseteq \mathcal{L} \setminus \mathcal{L}_i$ similarly for $\hat{\sigma}'$, then $\mathcal{L}' = \mathcal{L}''$. Therefore, $S = S'$ holds. Using the same reasoning as in Cases 1 and 2, $\mathrm{Cr}_{\hat{\sigma}'}^{\hat{F}'}(L) \geq \mathrm{Cr}_{\hat{\sigma}}^{\hat{F}}(L)$.

Case 4. The fourth case is when L appears between L_i^* and L_i^{**} in the permutation $\hat{\sigma}$, and the left boundary of S belongs to $\mathcal{L}_i$. Notice that the left boundary of S must either coincide with L_i^*, or appear to the right of it.

Assume first that $b_i = 0$, so we have replaced L_i^* with the line L_i^1, that lies to the left of L_i^*. Since no other lines of $\mathcal{L}_i$ appear in $\hat{\sigma}'$ until the original location of line L_i^{**}, it is easy to verify that the right boundary of S' is the same as the right boundary of S, and its left boundary is the line L_i^1, that is, we have pushed the left boundary to the left. In order to prove that $|\mathrm{Cr}_{\hat{\sigma}'}^{\hat{F}'}(L)| \geq |\mathrm{Cr}_{\hat{\sigma}}^{\hat{F}}(L)|$, we map every crossing $(p_1, p_2) \in \mathrm{Cr}_{\hat{\sigma}}^{\hat{F}}(L)$ to some crossing $(p_1', p_2') \in \mathrm{Cr}_{\hat{\sigma}'}^{\hat{F}'}(L)$, so that no two crossings of $\mathrm{Cr}_{\hat{\sigma}}^{\hat{F}}(L)$ are mapped to the same crossing of $\mathrm{Cr}_{\hat{\sigma}'}^{\hat{F}'}(L)$.

Consider any crossing $(p_1, p_2) \in \mathrm{Cr}_{\hat{\sigma}}^{\hat{F}}(L)$ (see Figure 4). We know that $p_1, p_2 \in S$, and they lie on opposite sides of L. We assume w.l.o.g. that p_1 lies to the left of L. Moreover, no point of $X \cap S$ lies between the row of p_1 and the row of p_2. It is however possible that (p_1, p_2) is not a crossing of $\mathrm{Cr}_{\hat{\sigma}'}(L)$, since by moving the left boundary of S to the left, we add more points to the strip, some of which may lie between the row of p_1 and the row of p_2. Let A be the set of all points that lie between the row of p_1 and the row of p_2 in S'. Notice that the points of A are not forbidden in $\hat{F}'$. Let $a \in A$ be the point of a whose row is closest to the row of p_2; if $A = \emptyset$, then we set $a = p_1$. Then (a, p_2) defines a crossing in $\mathrm{Cr}_{\hat{\sigma}'}(L)$, and, since neither point lies in $\hat{F}'$, $(a, p_2) \in \mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}(L)$. In this way, we map every crossing $(p_1, p_2) \in \mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}(L)$ to some crossing $(p'_1, p'_2) \in \mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}(L)$. It is easy to verify that no two crossings of $\mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}(L)$ are mapped to the same crossing of $\mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}(L)$. We conclude that $|\mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}(L)| \geq |\mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}(L)|$.

Lastly, assume that $b_i = 1$. Recall that the set of all points of X lying between L^*_i and $L^{q_i+1}_i$ is forbidden in $\hat{F}$ but not in $\hat{F}'$, and that we have replaced L^*_i with the line $L^{q_i+1}_i$, that lies to the right of L^*_i. Therefore, the right boundary of S remains the same, and the left boundary is pushed to the right. In order to prove that $|\mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}(L)| \geq |\mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}(L)|$, we show that every crossing $(p_1, p_2) \in \mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}(L)$ belongs to $\mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}(L)$. Indeed, consider any crossing $(p_1, p_2) \in \mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}(L)$. We know that $p_1, p_2 \in S$, and they lie on opposite sides of L. We assume w.l.o.g. that p_1 lies to the left of L. Since p_1 cannot be a forbidden point, it must lie to the right of $L^{q_i+1}_i$. Moreover, no point of $X \cap S$ lies between the row of p_1 and the row of p_2. It is now easy to verify that (p_1, p_2) is also a crossing in $\mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}(L)$.

- **Case 5.** The fifth case happens when L appears between L^*_i and L^{**}_i in the permutation $\hat{\sigma}$, and the right boundary of S belongs to $\mathcal{L}_i$. This case is symmetric to the fourth case and is analyzed similarly.

□

It now remains to analyze the crossings of the lines in $\mathcal{L}_i$. We do so in the following two claims. The first claim shows that switching L^*_i with L^1_i or $L^{q_i+1}_i$ does not decrease the number of crossings.

Claim 3.10. *If $b = 0$, then $|\mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}(L^1_i)| \geq |\mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}(L^*_i)|$; if $b = 1$, then $|\mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}(L^{q_i+1}_i)| \geq |\mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}(L^*_i)|$.*

Proof. Assume first that $b = 0$, so we have replaced L^*_i with L^1_i in the permutation. As before, we let $v \in V(T(\hat{\sigma}))$ be the vertex to which L^*_i belongs, and we let $S = S(v)$ be the corresponding strip. Similarly, we define $v' \in V(T(\hat{\sigma}'))$ and $S' = S(v')$ with respect to line L^1_i and permutation $\hat{\sigma}'$. Notice that, until the appearance of L^*_i in $\hat{\sigma}$, the two permutations are identical. Therefore, $S = S'$ must hold. Recall also that all points of X that lie between L^*_i and L^1_i are forbidden in

$\hat{F}$, but not in $\hat{F}'$. In order to show that $|\mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}(L^1_i)| \geq |\mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}(L^*_i)|$, it is enough to show that every crossing $(p_1, p_2) \in \mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}(L^*_i)$ also lies in $\mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}(L^1_i)$.

Consider now some crossing $(p_1, p_2) \in \mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}(L^*_i)$. Recall that one of p_1, p_2 must lie to the left of L^*_i and the other to the right of it, with both points lying in S. Assume w.l.o.g. that p_1 lies to the left of L^*_i. Since $p_1 \notin \hat{F}$, it must lie to the left of L^1_i. Moreover, no point of $X \cap S$ may lie between the row of p_1 and the row of p_2. It is then easy to verify that (p_1, p_2) is also a crossing in $\mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}(L^1_i)$, and so $|\mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}(L^1_i)| \geq |\mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}(L^*_i)|$.

The second case, when $b = 1$, is symmetric. □

Lastly, we show that for all lines $L \in \mathcal{L}_i \setminus \{L^*_i\}$, their total contribution to $\mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}$ is small.

Claim 3.11. $\sum_{L \in \mathcal{L}_i \setminus \{L^*_i\}} |\mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}(L)| \leq \mathsf{WB}(X_i) + O(|X_i|)$.

Assume first that the claim is correct. We have shown so far that the total contribution of all lines in $\mathcal{L}_i \setminus \{L^*_i\}$ to $\mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}$ is at most $\mathsf{WB}(X_i) + O(|X_i|)$; that the contribution of one of the lines $L^1_i, L^{q_i+1}_i$ to $\mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}$ is at least as large as the contribution of L^*_i to $\mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}$; and that for every line $L \notin \mathcal{L}_i$, its contribution to $\mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}$ is at least as large as its contribution to $\mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}$. It then follows that $|\mathrm{Cr}^{\hat{F}'}_{\hat{\sigma}'}| \geq |\mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}| - \mathsf{WB}(X_i) + O(|X_i|)$, and so $\mathsf{WB}^{F_{i+1}}_{\sigma_{i+1}}(X) \geq \mathsf{WB}^{F_i}_{\sigma_i}(X) - \mathsf{WB}(X_i) + O(|X_i|)$. Therefore, in order to prove Lemma 3.8, it is now enough to prove Claim 3.11.

Proof. (Of Claim 3.11) Consider some line $L \in \mathcal{L}_i \setminus \{L^*_i\}$, and let $v \in V(T(\hat{\sigma}))$ be the vertex to which L belongs. Notice that L appears in $\hat{\sigma}$ after L^*_i. Therefore, if $S = S(v)$ is the strip that L partitioned, then at least one of the boundaries of S lies in $\mathcal{L}_i$. If exactly one boundary of S lies in $\mathcal{L}_i$, then we say that S is an *external strip*; otherwise, we say that S is an *internal* strip. Consider now some crossing $(p, p') \in \mathrm{Cr}_{\hat{\sigma}}(S)$. Since $L \in \mathcal{L}_i$, and at least one boundary of S lies in $\mathcal{L}_i$, at least one of the points p, p' must belong to X_i. If exactly one of p, p' lies in X_i, then we say that (p, p') is a type-1 crossing; otherwise it is a type-2 crossing. Notice that, if S is an internal strip, then only type-2 crossings of L are possible. We now bound the total number of type-1 and type-2 crossings separately, in the following two observations.

Observation 3.12. The total number of type-2 crossings in $\bigcup_{L \in \mathcal{L}_i \setminus \{L^*_i\}} \mathrm{Cr}_{\hat{\sigma}}(L)$ is at most $\mathsf{WB}(X_i)$.

Proof. Permutation $\hat{\sigma}$ of the lines in $\mathcal{L}$ naturally induces a permutation $\hat{\sigma}_i$ of the lines in $\mathcal{L}_i$. The number of type-2 crossings charged to all lines in $\mathcal{L}_i$ is then at most $\mathsf{WB}_{\hat{\sigma}_i}(X_i) \leq \mathsf{WB}(X_i)$. □

Observation 3.13. The total number of type-1 crossings in $\bigcup_{L \in \mathcal{L}_i \setminus \{L^*_i\}} \mathrm{Cr}_{\hat{\sigma}}(L) \leq O(|X_i|)$.

Proof. Consider a line $L \in \mathcal{L}_i \setminus \{L^*_i\}$, and let S be the strip that it splits. Recall that, if there are any type-1 crossings in $\mathrm{Cr}_{\hat{\sigma}}(L)$, then S must be an external strip. Line L partitions S into two new strips, that we denote by S' and S''. Notice that exactly one of these strips (say S') is an internal strip, and the other strip is external. Therefore, the points of $X_i \cap S'$ will never participate in type-1 crossings again. Recall that, from Observation 2.8, the total number of crossings in $\mathrm{Cr}_{\hat{\sigma}}(L)$ is bounded by $2|S' \cap X_i|$. We say that the points of $S' \cap X_i$ pay for these crossings. Since every point of X_i will pay for a type-1 crossing at most once, we conclude that the total number of type-1 crossings in $\bigcup_{L\in\mathcal{L}_i\setminus\{L_i^*\}} \mathrm{Cr}_{\hat{\sigma}}(L)$ is bounded by $2|X_i|$. □

We conclude that the total number of all crossings in $\bigcup_{L\in\mathcal{L}_i\setminus\{L_i^*\}} \mathrm{Cr}_{\hat{\sigma}}(L)$ is at most $\mathsf{WB}(X_i)+O(|X_i|)$. Since, for every line L, $\mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}(L) \subseteq \mathrm{Cr}_{\hat{\sigma}}(L)$, we get that $\sum_{L\in\mathcal{L}_i\setminus\{L_i^*\}} |\mathrm{Cr}^{\hat{F}}_{\hat{\sigma}}(L)| \leq \mathsf{WB}(X_i) + O(|X_i|)$. □

□

To summarize, we have transformed a permutation σ of $\mathcal{L}$ into a permutation σ' of $\mathcal{L}$, and we have shown that $\mathsf{WB}_{\sigma'}(X) \geq \mathsf{WB}(X)/4 - 2\sum_{i=1}^{k} \mathsf{WB}(X_i) - O(|X|)$.

3.3.5 Transforming σ' into $\tilde{\sigma}$

In this final step, we transform the permutation σ' of $\mathcal{L}$ into a permutation $\tilde{\sigma}$ of $\tilde{\mathcal{L}}$, and we will show that $\mathsf{WB}_{\tilde{\sigma}}(\tilde{X}) \geq \mathsf{WB}_{\sigma'}(X) - |X|$.

The transformation is straightforward. Consider some block $\mathcal{B}_i$, and the corresponding set $\mathcal{L}_i \subseteq \mathcal{L}$ of lines. Recall that the lines in $\mathcal{L}_i$ are indexed $L_i^1, \dots, L_i^{q_i+1}$ in this left-to-right order, where L_i^1 appears to the left of the first column of $\mathcal{B}_i$, and $L_i^{q_i+1}$ appears to the right of the last column of $\mathcal{B}_i$. Recall also that, in the current permutation σ', one of the following happens: either (i) line L_i^1 appears in the permutation first, and lines $L_i^{q_i+1}, L_i^2, \dots, L_i^q$ appear at some later point consecutively in this order; or (ii) line $L_i^{q_i+1}$ appears in the permutation first, and lines $L_i^1, L_i^2, \dots, L_i^q$ appear somewhere later in the permutation consecutively in this order. Therefore, for all $2 \leq j \leq q$, line L_i^j separates a strip whose left boundary is L_i^{j-1} and right boundary is $L_i^{q_i+1}$. It is easy to see that the cost of each such line L_i^j in permutation σ' is bounded by the number of points of X that lie on the unique active column that appears between L_i^{j-1} and L_i^j. The total cost of all such lines is then bounded by $|X_i|$.

Let $\tilde{\sigma}^*$ be a sequence of lines obtained from σ' by deleting, for all $1 \leq i \leq k$, all lines $L_i^2, \dots, L_i^q$ from it. Then $\tilde{\sigma}^*$ naturally defines a permutation $\tilde{\sigma}$ of the set $\tilde{\mathcal{L}}$ of vertical lines. Moreover, from the above discussion, the total contribution of all deleted lines to $\mathsf{WB}_{\sigma'}(X)$ is at most $|X|$, so $\mathsf{WB}_{\tilde{\sigma}}(\tilde{X}) \geq \mathsf{WB}_{\sigma'}(X) - |X| \geq \mathsf{WB}(X)/4 - 2\sum_i \mathsf{WB}(X_i) - O(|X|)$. We conclude that $\mathsf{WB}(\tilde{X}) \geq \mathsf{WB}_{\tilde{\sigma}}(\tilde{X}) \geq \mathsf{WB}(X)/4 - 2\sum_i \mathsf{WB}(X_i) - O(|X|)$, and $\mathsf{WB}(X) \leq 4\mathsf{WB}(\tilde{X}) + 8\sum_i \mathsf{WB}(X_i) + O(|X|)$.

4 Separation results for the strong Wilber bound

In this section we present our negative results, proving Theorem 1.1, and extend it to obtain a separation result between the first and second Wilber bounds.

4.1 Basic tools

Our construction combines known input sequences and their properties, some of which have been proved in the standard tree view of binary search trees. We discuss these facts in the geometric context.

4.1.1 Monotonically increasing sequence

We say that an input set X of points is *monotonically increasing* iff X is a permutation, and moreover for every pair $p, p' \in X$ of points, if $p.x < p'.x$, then $p.y < p'.y$ must hold. It is well known that the value of the optimal solution of monotonically increasing sequences is low, and we exploit this fact in our negative results.

Observation 4.1. If X is a monotonically increasing set of points, then $\mathsf{OPT}(X) \leq |X| - 1$.

Proof. We order points in X based on their x-coordinates as $X = \{p_1, \ldots, p_m\}$ such that $p_1.x < p_2.x < \ldots < p_m.x$. For each $i = 1, \ldots, m-1$ we define $q_i = ((p_i).x, (p_{i+1}).y)$ and the set $Y = \{q_1, \ldots, q_{m-1}\}$. It is easy to verify that Y is a feasible solution for X. □

4.1.2 Bit reversal sequence (BRS)

The bit-reversal sequence, first described by Wilber [29], is a family of explicit input sequences whose optimal value is asymptotically largest possible, that is, $\mathsf{OPT}(X) = \Omega(|X| \log |X|)$. The original sequence was described in the language of binary representation of strings. Here we use the geometric variant of BRS, which is more convenient for our analysis.

Let $i \geq 0$ be an integer and $\mathcal{R} \subseteq \mathbb{N}$ and $\mathcal{C} \subseteq \mathbb{N}$ be subsets of active rows and columns such that $|\mathcal{R}| = |\mathcal{C}| = 2^i$. The level-$i$ bit-reversal instance $\mathsf{BRS}(i, \mathcal{R}, \mathcal{C})$ contains 2^i points whose sets of active rows and columns are exactly $\mathcal{R}$ and $\mathcal{C}$ respectively. The instances are defined inductively.

The level-0 instance $\mathsf{BRS}(0, \{C\}, \{R\})$, containing a single point at the intersection of row R and column C. Assume now that we have defined, for all $1 \leq i' \leq i$, and any sets $\mathcal{R}', \mathcal{C}'$ of $2^{i'}$ integers, the corresponding instance $\mathsf{BRS}(i', \mathcal{R}', \mathcal{C}')$. We define instance $\mathsf{BRS}(i+1, \mathcal{R}, \mathcal{C})$, where $|\mathcal{R}| = |\mathcal{C}| = 2^{i+1}$, as follows.

Consider the columns in $\mathcal{C}$ in their natural left-to-right order, and define $\mathcal{C}_{left}$ to be the first 2^i columns and $\mathcal{C}_{right} = \mathcal{C} \setminus \mathcal{C}_{left}$. Denote $\mathcal{R} = \{R_1, \ldots, R_{2^{i+1}}\}$, where the rows are indexed in their natural bottom to top order, and let $\mathcal{R}_{even} = \{R_2, R_4, \ldots, R_{2^{i+1}}\}$ and $\mathcal{R}_{odd} =$

$\{R_1, R_3, \ldots, R_{2^{i+1}-1}\}$ be the sets of all even-indexed and all odd-indexed rows, respectively. Notice that $|\mathcal{C}_{left}| = |\mathcal{C}_{right}| = |\mathcal{R}_{even}| = |\mathcal{R}_{odd}| = 2^i$. The instance $\mathsf{BRS}(i+1, \mathcal{R}, \mathcal{C})$ is defined to be $\mathsf{BRS}(i, \mathcal{R}_{odd}, \mathcal{C}_{left}) \cup \mathsf{BRS}(i, \mathcal{R}_{even}, \mathcal{C}_{right})$. See Figure 5 for an illustration.

It is well-known [29] that, if X is a bit-reversal sequence on n points, then $\mathsf{OPT}(X) \geq \Omega(n \log n)$.

Claim 4.2. *Let $X = \mathsf{BRS}(i, \mathcal{C}, \mathcal{R})$, for any $i \geq 0$ and any sets $\mathcal{C}$ and $\mathcal{R}$ of columns and rows, respectively, with $|\mathcal{R}| = |\mathcal{C}| = 2^i$. Then $|X| = 2^i$, and $\mathsf{OPT}(X) \geq \frac{\mathsf{WB}(X)}{2} \geq \Omega(|X| \log |X|)$.*

Next, we present two additional technical tools that we use in our construction.

4.1.3 Exponentially spaced columns

Recall that we defined the bit reversal instance $\mathsf{BRS}(\ell, \mathcal{R}, \mathcal{C})$, where $\mathcal{R}$ and $\mathcal{C}$ are sets of 2^ℓ rows and columns, respectively, that serve in the resulting instance as the sets of active rows and columns; the instance contains $n = 2^\ell$ points. In the Exponentially-Spaced BRS instance $\mathsf{ES\text{-}BRS}(\ell, \mathcal{R})$, we are still given a set $\mathcal{R}$ of 2^ℓ rows that will serve as active rows in the resulting instance, but we define the set $\mathcal{C}$ of columns in a specific way. For an integer i, let C_i be the column whose x-coordinate is i and $\mathcal{C}$ contain, for each $0 \leq i < 2^\ell$, the column C_{2^i}. Denoting $N = 2^n = 2^{2^\ell}$, the x-coordinates of the columns in $\mathcal{C}$ are $\{1, 2, 4, 8, \ldots, N/2\}$. The instance is then defined to be $\mathsf{BRS}(\ell, \mathcal{R}, \mathcal{C})$ for this specific set $\mathcal{C}$ of columns. Notice that the instance contains $n = \log N = 2^\ell$ input points.

It is easy to see that any point set $X = \mathsf{ES\text{-}BRS}(\ell, \mathcal{R})$ satisfies $\mathsf{OPT}(X) = \Omega(n \log n)$. We remark that this idea of exponentially spaced columns is inspired by the instance used by Iacono [18] to prove a gap between the weak WB-1 bound and $\mathsf{OPT}(X)$. However, Iacono's instance is tailored to specific partitioning tree T, and it is clear that there is another partitioning tree T' with $\mathsf{OPT}(X) = \Theta(WB_{T'}(X))$. Therefore, this instance does not give a separation result for the strong WB-1 bound, and in fact it does not provide negative results for the weak WB-1 bound when the input point set is a permutation.

4.1.4 Cyclic shift of columns

Suppose we are given a point set X, and let $\mathcal{C} = \{C_0, \ldots, C_{N-1}\}$ be any set of columns indexed in their natural left-to-right order, such that all points of X lie on columns of $\mathcal{C}$ (but some columns may contain no points of X). Let $0 \leq s < N$ be any integer. We denote by X^s a cyclic shift of X by s units with respect to $\mathcal{C}$, obtained as follows. For every point $p \in X$ on column C_j, we add a new point p^s to X^s, that lies on the same row as p and on column $C_{(j+s) \mod N}$. In other words, we shift the point p by s steps to the right (with respect to $\mathcal{C}$) in a circular manner. Equivalently, we move the last s columns of $\mathcal{C}$ to the beginning of the instance. The following claim shows that the value of the optimal solution does not decrease significantly in the shifted instance.

Claim 4.3. *Let X be any point set that is a semi-permutation. Let $0 \le s < N$ be a shift value, and let $X' = X^s$ be the instance obtained from X by a cyclic shift of its points by s units to the right. Then* $\mathsf{OPT}(X') \ge \mathsf{OPT}(X) - |X|$.

Proof. Let Y' be an optimal canonical solution to instance X'. We partition Y' into two subsets: set Y'_1 consists of all points lying on the first s columns with integral coordinates, and set Y'_2 consists of all points lying on the remaining columns. We also partition the points of X' into two subsets X'_1 and X'_2 similarly. Notice that $X'_1 \cup Y'_1$ must be a satisfied set of points, and similarly, $X'_2 \cup Y'_2$ is a satisfied set of points. Our goal is to use these sets to construct a feasible solution for X of size $|X| + |Y'_1| + |Y'_2| = |X| + \mathsf{OPT}(X')$.

Next, we partition the set X of points into two subsets: set X_1 contains all points lying on the last s columns with integral coordinates, and set X_2 contains all points lying on the remaining columns. Since X_1 and X_2 are simply horizontal shifts of the sets X'_1 and X'_2 of points, we can define a set Y_1 of $|Y'_1|$ points such that Y_1 is a canonical feasible solution for X_1, and we can define a set Y_2 for X_2 similarly. Let C be a column with a half-integral x-coordinate that separates X_1 from X_2 (that is, all points of X_1 lie to the right of C while all points of X_2 lie to its left.) We construct a new set Z of points, of cardinality $|X|$, such that $Y_1 \cup Y_2 \cup Z$ is a feasible solution to instance X. In order to construct the point set Z, for each point $p \in X$, we add a point p' with the same y-coordinate, that lies on column C, to Z. Notice that $|Z| = |X|$.

We claim that $Z \cup (Y_1 \cup Y_2)$ is a feasible solution for X, and this will complete the proof. Consider any two points $p, q \in Z \cup (Y_1 \cup Y_2) \cup X$ that are not aligned. Let B_1 and B_2 be the strips obtained from the bounding box B by partitioning it with column C, so that $X_1 \subseteq B_1$ and $X_2 \subseteq B_2$. If both p and q lie in the interior of the same strip, say B_1, we are done since set $X_1 \cup Y_1$ of points is satisfied. So, assume that one of the points (say p) lies in the interior of one of the strips (say B_1), while the other point either lies on C, or in the interior of B_2. Then $p \in X_1 \cup Y_1$ must hold. Moreover, since Y_1 is a canonical solution for X_1, point p lies on a row that is active for X_1. Therefore, some point $p' \in X_1$ lies on the same row (where possibly $p' = p$). But then a copy of p' that was added to the set Z and lies on the column C satisfies the pair (p, q). $\square$

4.1.5 Partial costs of WB-1 bound

We use simple facts about the Wilber bound. The following is a property of any balanced binary search trees.

Lemma 4.4. *For any semi-permutation X,* $\mathsf{WB}(X) \le 2\mathsf{OPT}(X) \le O(r(X) \log c(X))$.

Our analysis also uses partial costs of the WB-1 bound restricted to a subtree and a path.

Claim 4.5. *Consider a set X of points that is a semi-permutation, an ordering σ of the auxiliary columns in $\mathcal{L}$ and the corresponding partitioning tree $T = T(\sigma)$. Let $v \in V(T)$ be any vertex of the tree. Then the following hold:*

- *Let T_v be the subtree of T rooted at v. Then*

$$\sum_{u \in V(T_v)} \mathsf{cost}(u) = \mathsf{WB}_{T_v}(X \cap S(v)) \leq O(N(v) \log(N(v)))$$

- *Let u be any descendant vertex of v, and let P be the unique path in T connecting u to v. Then $\sum_{z \in V(P)} \mathsf{cost}(z) \leq 2N(v)$.*

Proof. The first assertion follows from the definition of the weak WB-1 bound and Lemma 4.4. We now prove the second assertion. Denote $P = (v = v_1, v_2, \ldots, v_k = u)$. For all $1 < i \leq k$, we let v'_i be the unique sibling of the vertex v_i in the tree T. We also let X_i be the set of points of X that lie in the strip $S(v'_i)$, and we let X' be the set of all points of X that lie in the strip $S(v_k)$. It is easy to verify that $X_2, \ldots, X_k, X'$ are all mutually disjoint (since the strips $\{S(v'_i)\}_{i=2}^{q}$ and $S(v_k)$ are disjoint), and that they are contained in $X \cap S(v)$. Therefore, $\sum_{i=2}^{q} |X_i| + |X'| \leq N(v)$.

From Observation 2.8, for all $1 \leq i < k$, $\mathsf{cost}(v_i) \leq 2N(v'_i) = 2|X_i|$, and $\mathsf{cost}(v_k) \leq 2N(v_k) = 2|X'|$. Therefore, $\sum_{z \in V(P)} \mathsf{cost}(z) \leq 2 \sum_{i=2}^{q} |X_i| + 2|X'| \leq 2N(v)$. □

4.2 Construction of the bad instance

We construct two instances: instance $\hat{X}$ on N^* points, that is a semi-permutation (but is somewhat easier to analyze), and instance X^* in N^* points, which is a permutation; the analysis of instance X^* heavily relies on the analysis of instance $\hat{X}$. We will show that the optimal solution value of both instances is $\Omega(N^* \log\log N^*)$, but the cost of the Wilber Bound is at most $O(N^* \log\log\log N^*)$. Our construction uses the following three parameters. We let $\ell \geq 1$ be an integer, and we set $n = 2^{\ell}$ and $N = 2^n$.

4.2.1 First instance

We now construct our first instance $\hat{X}$, which is a semi-permutation containing N columns. Intuitively, we create N instances $X^0, X^1, \ldots, X^{N-1}$, where instance X^s is an exponentially-spaced BRS instance that is shifted by s units. We then stack these instances on top of one another in this order.

Formally, for all $0 \leq j \leq N-1$, we define a set $\mathcal{R}_j$ of n consecutive rows with integral coordinates, such that the rows of $\mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_{N-1}$ appear in this bottom-to-top order. Specifically, set $\mathcal{R}_j$ contains all rows whose y-coordinates are in $\{jn+1, jn+2, \ldots, (j+1)n\}$.

For every integer $0 \leq s \leq N-1$, we define a set of points X^s, which is a cyclic shift of instance $\mathsf{ES\text{-}BRS}(\ell, \mathcal{R}_s)$ by s units. Recall that $|X^s| = 2^{\ell} = n$ and that the points in X^s appear on the rows in $\mathcal{R}_s$ and a set $\mathcal{C}_s$ of columns, whose x-coordinates are in $\{(2^j + s) \mod N : 0 \leq j < n\}$. We then let our final instance be $\hat{X} = \bigcup_{s=0}^{N-1} X^s$. From now on, we denote $N^* = |\hat{X}|$. Recall that $|N^*| = N \cdot n = N \log N$.

Observe that the number of active columns in $\hat{X}$ is N. Since the instance is symmetric and contains $N^* = N \log N$ points, every column contains exactly $\log N$ points. Each row contains exactly one point, so $\hat{X}$ is a semi-permutation. (See Figure 5 for an illustration).

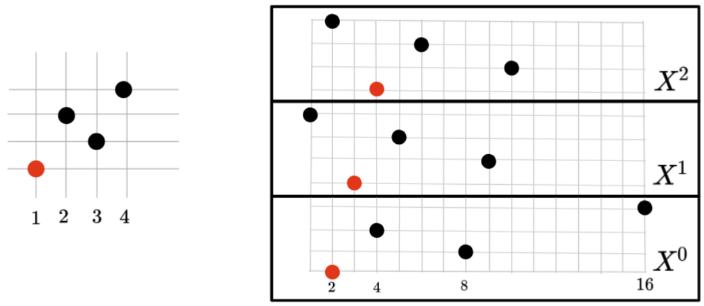


Figure 5: An illustration of our construction. The figure on the left shows the instance $\mathsf{BRS}(2, [4], [4])$. The figure on the right combines three copies X^0, X^1, X^2 of the corresponding exponentially-spaced instance, with horizontal shifts of 0, 1, and 2, respectively. The red points are shifted copies of the same point in different sub-instances.

Lastly, we need the following bound on the value of the optimal solution of instance $\hat{X}$.

Observation 4.6. $\mathsf{OPT}(\hat{X}) = \Omega(N^* \log\log N^*)$

Proof. From Claims 4.2 and 4.3, for each $0 \le s \le N-1$, each sub-instance X^s has $\mathsf{OPT}(X^s) \ge \Omega(n \log n) = \Omega(\log N \log\log N)$. Therefore, $\mathsf{OPT}(\hat{X}) \ge \sum_{s=0}^{N-1} \mathsf{OPT}(X^s) = \Omega(N \log N \log\log N) = \Omega(N^* \log\log N^*)$ (we have used the fact that $N^* = N \log N$). □

4.2.2 Second instance

We now construct our second and final instance, X^*, that is a permutation. In order to do so, we start with the instance $\hat{X}$, and, for every active column C of $\hat{X}$, we create $n = \log N$ new columns (that we view as copies of C), $C^1, \ldots, C^{\log N}$, which replace the column C. We denote this set of columns by $\mathcal{B}(C)$, and we refer it as the *block of columns representing* C. Recall that the original column C contains $\log N$ input points of $\hat{X}$. We place each such input point on a distinct column of $\mathcal{B}(C)$, so that the points form a monotonically increasing sequence (see the definition in Section 4.1). This completes the definition of the final instance X^*. We obtain the following immediate bound on the optimal solution cost of instance X^*.

Claim 4.7. $\mathsf{OPT}(X^*) \ge \mathsf{OPT}(\hat{X}) = \Omega(N^* \log\log N^*)$.

Next, we proceed to prove the following theorem.

Theorem 4.8. $\mathsf{WB}(\hat{X}) \le O(N^* \log\log\log N^*)$.

Recall again the instance $\hat{X}$. Recall that it consists of N instances $X^0, X^1, \ldots, X^{N-1}$ that are stacked on top of each other vertically in this order. We rename these instances as $X_1, X_2, \ldots, X_N$, so X_j is exactly $\mathsf{ES\text{-}BRS}(\log N)$, that is shifted by $(j-1)$ units to the right. Recall that $|\hat{X}| = N^* = N \log N$, and each instance X_s contains exactly $\log N$ points. We denote by $\mathcal{C}$ the set of N columns, whose x-coordinates are $1, 2, \ldots, N$. All points of $\hat{X}$ lie on the columns of $\mathcal{C}$. For convenience, for $1 \le j \le N$, we denote by C_j the column of $\mathcal{C}$ whose x-coordinate is j.

Let σ be any ordering of the auxiliary columns in $\mathcal{L}$, and let $T = T(\sigma)$ be the corresponding partitioning tree. It is enough to show that, for any such ordering σ, the value of $\mathsf{WB}_\sigma(\hat{X})$ is bounded by $O(N^* \log\log\log N^*)$.

The total costs of the bound is divided into two parts as follows. Recall that $WB_\sigma(\hat{X})$ is the sum, over all vertices $v \in V(T)$, of $\mathsf{cost}(v)$. If v is a leaf vertex, then $\mathsf{cost}(v) = 0$. Otherwise, let $L = L(v)$ be the line of $\mathcal{L}$ that v owns. Index the points in $X \cap S(v)$ by $q_1, \ldots, q_z$ in their bottom-to-top order. A consecutive pair (q_j, q_{j+1}) of points is a *crossing* iff they lie on different sides of $L(v)$. We distinguish between the two types of crossings that contribute towards $\mathsf{cost}(v)$. We say that the crossing (q_j, q_{j+1}) is of *type*-1 if both q_j and q_{j+1} belong to the same shifted instance X_s for some $0 \le s \le N-1$. Otherwise, they are of type-2. Note that, if (q_j, q_{j+1}) is a crossing of type 2, with $q_j \in X_s$ and $q_{j+1} \in X_{s'}$, then s, s' are not necessarily consecutive integers, as it is possible that for some indices s'', $X_{s''}$ has no points that lie in the strip $S(v)$. We now let $\mathsf{cost}_1(v)$ be the total number of type-1 crossings of $L(v)$, and $\mathsf{cost}_2(v)$ the total number of type-2 crossings. Note that $\mathsf{cost}(v) = \mathsf{cost}_1(v) + \mathsf{cost}_2(v)$. We also define $\mathsf{cost}_1(\sigma) = \sum_{v \in V(T)} \mathsf{cost}_1(v)$ and $\mathsf{cost}_2(\sigma) = \sum_{v \in V(T)} \mathsf{cost}_2(v)$. Clearly, $\mathsf{WB}_\sigma(\hat{X}) = \mathsf{cost}_1(\sigma) + \mathsf{cost}_2(\sigma)$. We prove the following two theorems.

Theorem 4.9. *For every ordering σ of the auxiliary columns in $\mathcal{L}$, $\mathsf{cost}_1(\sigma) \le O(N^* \log\log\log N^*)$.*

Theorem 4.10. *For every vertex $v \in V(T)$, $\mathsf{cost}_2(v) \le O(\log N) + O(\mathsf{cost}_1(v))$.*

We prove these theorems in Section 4.3 and 4.4. The latter implies that $\mathsf{cost}_2(\sigma) \le O(\mathsf{cost}_1(\sigma)) + O(|V(T)| \cdot \log N) = O(N^* \log\log\log N^*) + O(N \log N) = O(N^* \log\log\log N^*)$. Combining the two theorems together completes the proof of Theorem 4.8.

4.2.3 Upper bounding $\mathsf{WB}(X^*)$

We argue that $\mathsf{WB}(X^*) = O(N^* \log\log\log N^*)$, completing the proof of Theorem 1.1. Recall that instance X^* is obtained from instance $\hat{X}$ by replacing every active column C of X^* with a block $\mathcal{B}(C)$ of columns, and then placing the points of C on the columns of $\mathcal{B}(C)$ so that they form a monotone increasing sequence, while preserving their y-coordinates. The resulting collection of all blocks $\mathcal{B}(C)$ partitions the set of all active columns of X^*. We denote this set of blocks by $\mathcal{B}_1, \ldots, \mathcal{B}_N$. The idea is to use Theorem 3.6 in order to bound $\mathsf{WB}(X^*)$.

Consider a set of lines $\mathcal{L}'$ (with half-integral x-coordinates) that partition the bounding box B into N strips, where the ith strip contains the block $\mathcal{B}_i$ of columns, so $|\mathcal{L}'| = (N-1)$. We consider a split of instance X^* by $\mathcal{L}'$: This gives us a collection of strip instances $\{X_i^*\}_{1\le i\le N}$ and the compressed instance $\widetilde{X}^*$. Notice that the compressed instance is precisely $\hat{X}$, and each strip instance X_i^* is a monotone increasing point set.

Since each strip instance X_i^* is monotonously increasing, from Observation 4.1 and Claim 2.7, for all i, $\mathsf{WB}(X_i^*) \le O(\mathsf{OPT}(X_i^*)) \le O(|X_i^*|)$. From Theorem 3.6, we then get that: $\mathsf{WB}(X^*) \le 4\mathsf{WB}(\hat{X}) + 8\sum_i \mathsf{WB}(X_i^*) + O(|X^*|) \le 4\mathsf{WB}(\hat{X}) + O(|X^*|) \le O(N^* \log\log\log N^*)$.

4.3 Bounding type-1 crossings

The goal of this subsection is to prove Theorem 4.9.

We prove this theorem by a probabilistic argument. Consider the following experiment. Fix the permutation σ of $\mathcal{L}$. Pick an integer $s \in \{0, \dots, N-1\}$ uniformly at random, and let X be the resulting instance X_s. This random process generates an input X containing $n = \log N$ points. Equivalently, let $p_1, p_2, \dots, p_{\log N}$ be the points in $\mathsf{BRS}(\ell)$ ordered from left to right. Once we choose a random shift s, we move these points to columns in $\mathcal{C}_s = \{2^j + s \mod N\}$, where point p_j would be moved to x-coordinate $2^j + s \mod N$. Therefore, in the analysis, we view the location of points $p_1, \dots, p_{\log N}$ as random variables.

We denote by $\mu(\sigma)$ the expected value of $\mathsf{WB}_\sigma(X)$, over the choices of the shift s. The following observation is immediate, and follows from the fact that the final instance $\hat{X}$ contains every instance X_s for all shifts $s \in \{0, \dots, N-1\}$.

Observation 4.11. $\mathsf{cost}_1(\sigma) = N \cdot \mu(\sigma)$

Therefore, in order to prove Theorem 4.9, it is sufficient to show that, for every fixed permutation σ of $\mathcal{L}$, $\mu(\sigma) \le O(\log N \log\log\log N)$ (recall that $N^* = N \log N$).

We assume from now on that the permutation σ (and the corresponding partitioning tree T) is fixed, and we analyze the expectation $\mu(\sigma)$. Let $v \in V(T)$. We say that $S(v)$ is a *seam strip* iff point p_1 lies in the strip $S(v)$. We say that $S(v)$ is a *bad* strip (or that v is a bad node) if the following two conditions hold: (i) $S(v)$ is not a seam strip; and (ii) $S(v)$ contains at least $100 \log\log N$ points of X. Let $\mathcal{E}(v)$ be the bad event that $S(v)$ is a bad strip.

Claim 4.12. *For every vertex $v \in V(T)$,* $\mathbf{Pr}[\mathcal{E}(v)] \le \frac{8\,\mathrm{width}(S(v))}{N \log^{100} N}$.

Proof. Fix a vertex $v \in V(T)$. For convenience, we denote $S(v)$ by S. Let s be the random integer chosen by the algorithm and let $X_s = X$ be the resulting point set. Assume that S is a bad strip, and let L be the vertical line that serves as the left boundary of S. Let p_j be the point of X_s that lies to the left of L, and among all such points, we take the one closest to L. Recall that for each $1 \le j < \log N$, there are $2^j - 1$ columns of $\mathcal{C}$ that lie between the column of p_j and the column of p_{j+1}.

If S is a bad strip, then it must contain points $p_{j+1}, p_{j+2}, \ldots, p_{j+q}$, where $q = 100 \log\log N$. Therefore, the number of columns of C in strip S is at least 2^{j+q-2}, or, equivalently, $\mathrm{width}(S) \geq 2^{j+q}/4 \geq (2^j \log^{100} N)/4$. In particular, $2^j \leq 4\,\mathrm{width}(S)/\log^{100} N$.

Therefore, in order for S to be a bad strip, the shift s must be chosen in such a way that the point p_j, that is the rightmost point of X_s lying to the left of L, has $2^j \leq 4\,\mathrm{width}(S)/\log^{100} N$. It is easy to verify that the total number of all such shifts s is bounded by $8\,\mathrm{width}(S)/\log^{100} N$.

In order to see this, consider an equivalent experiment, in which we keep the instance X_1 fixed, and instead choose a random shift $s \in \{0, \ldots, N-1\}$ for the line L. For the bad event $\mathcal{E}(v)$ to happen, the line L must fall in the interval between x-coordinate 0 and x-coordinate $8\,\mathrm{width}(S)/\log^{100} N$. Since every integral shift s is chosen with the same probability $1/N$, the probability that $\mathcal{E}(v)$ happens is at most $\frac{8\,\mathrm{width}(S)}{N \log^{100} N}$. □

Consider now the partitioning tree T. We partition the vertices of T into $\log N + 1$ classes $Q_1, \ldots, Q_{\log N+1}$. A vertex $v \in V(T)$ lies in class Q_i iff $2^i \leq \mathrm{width}(S(v)) < 2^{i+1}$. Therefore, every vertex of T belongs to exactly one class.

Consider now some vertex $v \in V(T)$, and assume that it lies in class Q_i. We say that v is an *important vertex* for class Q_i iff no ancestor of v in the tree T belongs to class Q_i. Notice that, if u is an ancestor of v, and $u \in Q_j$, then $j \geq i$ must hold.

For each $1 \leq i \leq \log N + 1$, let U_i be the set of all important vertices of class Q_i.

Observation 4.13. For each $1 \leq i \leq \log N + 1$, $|U_i| \leq N/2^i$.

Proof. Since no vertex of U_i may be an ancestor of another vertex, the strips in $\{S(v) \mid v \in U_i\}$ are mutually disjoint, except for possibly sharing their boundaries. Since each strip has width at least 2^i, and we have exactly N columns, the number of such strips is bounded by $N/2^i$. □

Let $\mathcal{E}$ be the bad event that there is some index $1 \leq i \leq \log N + 1$, and some important vertex $v \in U_i$ of class Q_i, for which the event $\mathcal{E}(v)$ happens. Applying the Union Bound to all strips in $\{S(v) \mid v \in U_i\}$ and all indices $1 \leq i \leq \log N$, we obtain the following corollary of Claim 4.12.

Corollary 4.14. $\mathbf{Pr}[\mathcal{E}] \leq \frac{32}{\log^{99} N}$.

Proof. Fix some index $1 \leq i \leq \log N$. Recall that for every important vertex $v \in U_i$, the probability that the event $\mathcal{E}(v)$ happens is at most $\frac{8\,\mathrm{width}(S(v))}{N \log^{100} N} \leq \frac{8 \cdot 2^{i+1}}{N \log^{100} N} = \frac{16 \cdot 2^i}{N \log^{100} N}$. From Observation 4.13, $|U_i| \leq N/2^i$. From the union bound, the probability that event $\mathcal{E}(v)$ happens for any $v \in U_i$ is bounded by $\frac{16}{\log^{100} N}$. Using the union bound over all $1 \leq i \leq \log N + 1$, we conclude that $\mathbf{Pr}[\mathcal{E}] \leq \frac{32}{\log^{99} N}$. □

Lastly, we show that, if event $\mathcal{E}$ does not happen, then the cost of the Wilber Bound is sufficiently small.

Lemma 4.15. *Let $1 \le s \le N$ be a shift for which $\mathcal{E}$ does not happen. Then:*

$$WB_\sigma(X_s) \le O(\log N \log\log\log N).$$

Proof. Consider the partitioning tree $T = T(\sigma)$. We say that a vertex $v \in V(T)$ is a *seam vertex* iff $S(v)$ is a seam strip, that is, the point p_1 in instance X_s lies in $S(v)$. Clearly, the root of T is a seam vertex, and for every seam vertex v, exactly one of its children is a seam vertex. Therefore, there is a root-to-leaf path P that only consists of seam vertices, and every seam vertex lies on P. We denote the vertices of P by $v_1, v_2, \ldots, v_q$, where v_1 is the root of T, and v_q is a leaf. For $1 < i \le q$, we denote by v'_i the sibling of the vertex v_i. Note that all strips $S(v'_2), \ldots, S(v'_q)$ are mutually disjoint, except for possibly sharing boundaries, and so $\sum_{i=2}^{q} N(v'_i) \le |X_s| = \log N$. Moreover, from Claim 4.5, $\sum_{i=1}^{q} \mathsf{cost}(v_i) \le 2|X_s| = 2\log N$.

For each $1 < i \le q$, let T_i be the subtree of T rooted at the vertex v'_i. We prove the following claim:

Claim 4.16. *For all $1 < i \le q$, the total cost of all vertices in T_i is at most $O(N(v'_i)\log\log\log N)$.*

Assume first that the above claim is correct. Notice that every vertex of T that does not lie on the path P must belong to one of the trees T_i. The total cost of all vertices lying in all trees T_i is then bounded by $\sum_{i=2}^{q} O(N(v'_i)\log\log\log N) \le O(\log N \log\log\log N)$. Since the total cost of all vertices on the path P is bounded by $2\log N$, overall, the total cost of all vertices in T is bounded by $O(\log N \log\log\log N)$.

In order to complete the proof of Lemma 4.15, it now remains to prove Claim 4.16.

Proof. Claim 4.16 We fix some index $1 < i \le q$, and consider the vertex v'_i. If the parent v_{i-1} of v'_i belongs to a different class than v'_i, then v'_i must be an important vertex in its class. In this case, since we have assumed that Event $\mathcal{E}$ does not happen, $N(v'_i) \le O(\log\log N)$. From Claim 4.5, the total cost of all vertices in T_i is bounded by

$$\sum_{v \in V(T_i)} \mathsf{cost}(v) \le O(N(v'_i)\log(N(v'_i))) \le O(N(v'_i)\log\log\log N)$$

Therefore, we can assume from now on that v_{i-1} and v'_i both belong to the same class, that we denote by Q_j. Notice that, if a vertex v belongs to class Q_j, then at most one of its children may belong to class Q_j; the other child must belong to some class $Q_{j'}$ for $j' < j$, and it must be an important vertex in its class.

We now construct a path P_i in tree T_i iteratively, as follows. The first vertex on the path is v'_i. We then iteratively add vertices at to the end of path P_i one-by-one, so that every added vertex belongs to class Q_j. In order to do so, let v be the last vertex on the current path P_i. If some child u of v also lies in class Q_i, then we add u to the end of P_i and continue to the next iteration. Otherwise, we terminate the construction of the path P_i.

Denote the sequence of vertices on the final path P_i by $(v_i' = u_1, u_2, \dots, u_z)$; recall that every vertex on P_i belongs to class Q_j, and that path P_i is a sub-path of some path connecting v_i' to a leaf of T_i. Let Z be a set of vertices containing, for all $1 < z' \le z$ a sibling of the vertex $u_{z'}$, and additionally the two children of u_z (if they exist). Note that every vertex $x \in Z$ is an important vertex in its class, and, since we have assumed that Event $\mathcal{E}$ did not happen, $N(x) \le O(\log\log N)$. For every vertex $x \in Z$, we denote by T_x' the subtree of T rooted at x. From Claim 4.5, $\mathsf{cost}(T_x') \le O(N(x)\log(N(x))) = O(N(x)\log\log\log N)$.

Notice that all strips in $\{S(x) \mid x \in Z\}$ are disjoint from each other, except for possibly sharing a boundary. It is then easy to see that $\sum_{x\in Z} N(x) \le N(v_i')$. Therefore, altogether $\sum_{x\in Z}\mathsf{cost}(T_x') \le O(N(v_i')\log\log\log N))$.

Lastly, notice that every vertex of $V(T_i)$ either lies on P_i, or belongs to one of the trees T_x' for $x \in Z$. Since, from Claim 4.5, the total cost of all vertices on P_i is bounded by $N(v_i')$, altogether, the total cost of all vertices in T_i is bounded by $O(N(v_i')\log\log\log N))$. □

□

To summarize, if the shift s is chosen such that Event $\mathcal{E}$ does not happen, $WB_\sigma(X_s) \le O(\log N \log\log\log N)$. Assume now that the shift s is chosen such that Event $\mathcal{E}$ happens. From Corollary 4.14, the probability of this is at most $\mathbf{Pr}[\mathcal{E}] \le \frac{32}{\log^{99} N}$. Since $|X_s| = \log N$, from Corollary 4.4, $\mathsf{WB}_\sigma(X_s) \le |X_s|\log(|X_s|) \le \log N \log\log N$. Therefore, altogether, we get that $\mu(\sigma) \le O(\log N \log\log\log N)$, and $\mathsf{cost}_1(\sigma) = N \cdot \mu(\sigma) = O(N\log N \log\log\log N) = O(N^*\log\log\log N^*)$, as $N^* = N\log N$.

4.4 Bounding type-2 crossings

This subsection is dedicated to the proof of Theorem 4.10. We fix a vertex $v \in V(T)$, and we denote $S = S(v)$. We also let $L = L(v)$ be the vertical line that v owns. Our goal is to show that the number of type-2 crossings of L is bounded by $O(\mathsf{cost}_1(v)) + O(\log N)$.

Recall that instances $X_1, \dots, X_N$ are stacked on top of each other, so that the first $\log N$ rows with integral coordinates belong to X_1, the next $\log N$ rows belong to X_2, and so on. If we have a crossing (p, p'), where $p \in X_s$ and $p' \in X_{s'}$, then we say that the instances X_s and $X_{s'}$ are *responsible* for this crossings. Recall that p, p' may only define a crossing if they lie on opposite sides of the line L, and if no point of $\hat{X}$ lies in the strip S between the row of p and the row of p'. It is then clear that every instance X_s may be responsible for at most two type-2 crossings of L: one in which the second instance $X_{s'}$ responsible for the crossing has $s' < s$, and one in which $s' > s$.

We further partition the type-2 crossings into two sub-types. Consider a crossing (p, p'), and let $X_s, X_{s'}$ be the two instances that are responsible for it. If either of $X_s, X_{s'}$ contributes a type-1 crossing to the cost of L, then we say that (p, p') is a type-(2a) crossing; otherwise it is a type-(2b) crossing. Clearly, the total number of type-(2a) crossings is bounded by $O(\mathsf{cost}_1(v))$. It is now sufficient to show that the total number of all type-(2b) crossings is bounded by $O(\log N)$.

Consider now some type-(2b) crossing (p, p'), and let X_s and $X_{s'}$ be the two instances that are responsible for it, with $p \in X_s$. We assume that $s < s'$. Since neither instance contributes a crossing to $\mathsf{cost}_1(v)$, it must be the case that all points of $X_s \cap S$ lie to the left of L and all points of $X_{s'} \cap S$ lie to the right of L or vice versa. Moreover, if $s' > s + 1$, then for all $s < s'' < s'$, $X_{s''} \cap S = \emptyset$.

It would be convenient for us to collapse each of the instances $X_1, \ldots, X_N$ into a single row. In order to do so, for each $1 \leq s \leq N$, we replace all rows on which the points of X_s lie with a single row R_s. If some point of X_s lies on some column C, then we add a point at the intersection of R_s and C.

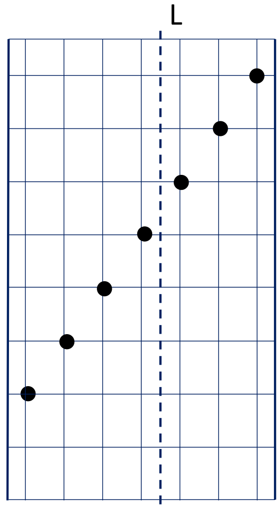
We say that a row R_s is *empty* if there are no input points in $R_s \cap S$. We say that it is a *neutral* row, if there are points in $R_s \cap S$ both to the left of L and to the right of L. We say that it is a *left row* if $R_s \cap S$ only contains points lying to the left of L, and we say that it is a *right row* if $R_s \cap S$ only contains points lying to the right of L.

If we now consider any type-(2b) crossing (p, p'), and the instances $X_s, X_{s'}$ that are responsible for it, with $s < s'$, then it must be the case that exactly one of the rows $R_s, R_{s'}$ is a left row, and the other is a right row. Moreover, if $s' > s + 1$, then every row lying between R_s and $R_{s'}$ is an empty row.

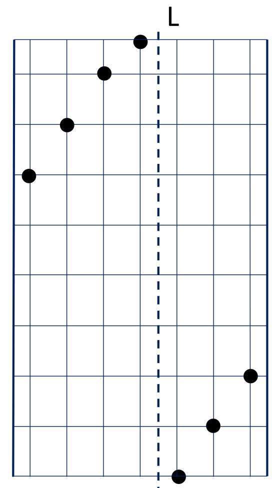
Let us denote the points in X_1 by $p_1, \ldots, p_{\log N}$, where for each $1 \leq i \leq \log N$, point p_i lies in column C_{2^i}. In each subsequent instance $X_2, X_3, \ldots$, the point is shifted by one unit to the right, so that in instance X_s it lies in column C_{2^i+s-1}; every column in $\mathcal{C}$ must contain exactly one copy of point p_i.

Consider now all copies of the point p_i that lie in the strip S. Let $\mathcal{R}_i$ be the set of rows containing these copies. Then two cases are possible: either (i) $\mathcal{R}_i$ is a contiguous set of rows, and the copies of p_i appear on $\mathcal{R}_i$ diagonally as an increasing sequence (the jth row of $\mathcal{R}_i$ contains a copy of p_i that lies in the jth column of $\mathcal{C}$ in the strip S); or $\mathcal{R}_i$ consists of two consecutive sets of rows; the first set, that we denote by $\mathcal{R}'_i$, contains R_1, and the second set, that we denote by $\mathcal{R}''_i$, contains the last row R_N. The copies of the point p_i also appear diagonally in $\mathcal{R}'_i$ and in $\mathcal{R}''_i$; in $\mathcal{R}''_i$ the first copy lies on the first column of $\mathcal{C}$ in S; in $\mathcal{R}'_i$ the last copy lies on the last column of $\mathcal{C}$ in S (see Figure 6).

We show that for each $1 \leq i \leq \log N$, there are at most four type-(2b) crossings of the line L in which a copy of p_i participates. Indeed, consider any type-(2b) crossing (p, p') in which a copy of p_i participates. We assume that the row of p lies below the row of p'. Assume first that both p and p' lie on rows of $\mathcal{R}_i$, and let R, R' be these two rows, with $p \in R$, $p' \in R'$. Recall that, in order for (p, p') to define a crossing, all input points that lie on $R \cap S$ must lie to the left of L, and all point points that lie on $R' \cap S$ must lie to the right of L, or the other way around. It is easy to verify (see Figure 6) that one of two cases must happen: either R contains a copy of p_i lying closest to L on its left, and R' contains a copy of p_i lying closest to L on its right; or R is the



(a) The consecutive set of rows on which the copies of p_i appear is denoted by $\mathcal{R}_i$.



(b) The two consecutive sets of rows on which the copies of p_i appear are $\mathcal{R}'_i$ (on the bottom) and $\mathcal{R}''_i$ (on the top).

Figure 6: Two patterns in which copies of p_i may appear on strip S.

last row of $\mathcal{R}'_i$, and R' is the first row of $\mathcal{R}''_i$. Therefore, only two such crossing, with $R, R' \in \mathcal{R}_i$ are possible.

Assume now that $R \in \mathcal{R}_i$ and $R' \notin \mathcal{R}_i$; recall that we assume that R' lies above R. Then all rows that lie between R and R' must be empty, so it is easy to verify that R must be the last row of $\mathcal{R}_i$ (or it must be the last row of $\mathcal{R}'_i$). In either case, at most one such crossing is possible.

Lastly, we assume that $R \notin \mathcal{R}_i$ and $R' \in \mathcal{R}_i$. The analysis is symmetric; it is easy to see that at most one such crossing is possible.

We conclude that for each $1 \le i \le \log N$, at most four type-(2b) crossings of the line L may involve copies of p_i, and so the total number of type-(2b) crossings of L is bounded by $O(\log N)$.

To summarize, we have shown that for every ordering σ of the auxiliary columns in $\mathcal{L}$, $\mathsf{cost}_1(\sigma), \mathsf{cost}_2(\sigma) \le O(N^* \log\log\log N^*)$, and so $\mathsf{WB}_\sigma(\hat{X}) = O(N^* \log\log\log N^*)$. Since $\mathsf{OPT}(\hat{X}) = \Omega(N^* \log\log N^*)$, we obtain a gap of $\Omega(\log\log N^* / \log\log\log N^*)$ between $\mathsf{OPT}(\hat{X})$ and $\mathsf{WB}(\hat{X})$.

4.5 Separating $\mathsf{WB}^{(2)}$ and WB

In this section, we extend our $\Omega(\frac{\log\log n}{\log\log\log n})$-factor separation between WB and OPT to a separation between WB and the *second Wilber bound* (denoted by $\mathsf{WB}^{(2)}$), which is defined below.[4]

[4]Wilber originally defined this bound based on the tree view. We use an equivalent geometric definition as discussed in [11, 18].

Let X be a set of m points that is a semi-permutation. Consider any point $p \in X$. The funnel of p, denoted by $\text{funnel}(X, p)$ is the set of all points $q \in X$, such that $q,y < p.y$, and $\square_{p,q}$ contains no point of $X \setminus \{p, q\}$. Denote $\text{funnel}(X, p) = \{a_1, a_2, \ldots, a_r\}$, where the points are indexed in the increasing order of their y-coordinates. Let $\text{alt}(X, p)$ be the number of indices $1 \le i < r$, such that a_i lies strictly to the left of p and a_{i+1} lies strictly to the right of p, or the other way around. The *second Wilber bound* is:

$$\mathsf{WB}^{(2)}(X) = m + \sum_{p \in X} \text{alt}(X, p).$$

The goal of this section is to prove the following:

Theorem 4.17. *For infinitely many integer n, there exists a point set X that is a permutation with $|X| = n$, such that $\mathsf{WB}^{(2)}(X) \ge \Omega(n \log\log n)$ but $\mathsf{WB}(X) \le O(n \log\log\log n)$.*

As it is known that $\mathsf{OPT}(X) \ge \mathsf{WB}^{(2)}(X)$ for any point set X [29], Theorem 4.17 is a stronger statement than Theorem 1.1. To prove Theorem 4.17, we use exactly the same permutation sequence X^* of size N^* that is constructed in Section 4.2. Since we already showed that $\mathsf{WB}(X^*) \le O(N^* \log\log\log N^*)$, it remains to show that $\mathsf{WB}^{(2)}(X^*) \ge \Omega(N^* \log\log N^*)$.

We use the following claim of Wilber [29].

Claim 4.18 ([29]). $\mathsf{WB}^{(2)}(\mathsf{BRS}(n)) = \Omega(n \log n)$ *for any n.*

We extend this bound to the cyclically shifted BRS in the following lemma.

Lemma 4.19. *For integers $n > 0, s$ with $0 \le s < n$, let X be the sequence obtained by performing a cyclic shift to $\mathsf{BRS}(n)$ by s units. Then $\mathsf{WB}^{(2)}(X) = \Omega(n \log n)$.*

Proof. Observe that, for any choice of s, there must exists a subsequence X' of X such that X' is a copy of $\mathsf{BRS}(n-1)$. It is shown in Lemma 6.2 of [22] that for any pair of sequences Z, Z' with $Z' \subseteq Z$, $\mathsf{WB}^{(2)}(Z') \le \mathsf{WB}^{(2)}(Z)$ holds. Therefore, we conclude that

$$\mathsf{WB}^{(2)}(X) \ge \mathsf{WB}^{(2)}(\mathsf{BRS}(n-1)) \ge \Omega(n \log n).$$

$\square$

Now, we are ready to bound $\mathsf{WB}^{(2)}(X^*)$.

Lemma 4.20. $\mathsf{WB}^{(2)}(X^*) = \Omega(N^* \log\log N^*)$.

Proof. Recall that $\hat{X}$ is the union of the sets $X^0, X^1, \ldots, X^{N-1}$ of points, where for all $0 \le s \le N-1$, set X^s is an exponentially-spaced BRS instance that is shifted by s units. From the definition of $\mathsf{WB}^{(2)}$, it is easy to see that $\mathsf{WB}^{(2)}(\hat{X}) \ge \sum_{s=0}^{N-1} \mathsf{WB}^{(2)}(X^s)$. This is since, for all $0 \le s \le N-1$, for ever point $p \in X^s$, funnel$(X^s, p) \subseteq$ funnel$(\hat{X}, p)$, and moreover alt$(\hat{X}, p) \geq$ alt(X^s, p). From Lemma 4.19, we get that $\mathsf{WB}^{(2)}(X^s) = \Omega(n \log n)$, where $n = |X^s| = \log N$. Therefore,

$$\mathsf{WB}^{(2)}(\hat{X}) \geq N \cdot \Omega(n \log n) = \Omega(N^* \log\log N^*).$$

Finally, recall that the sequence X^* is obtained from $\hat{X}$ by replacing each column C of $\hat{X}$ with a block $\mathcal{B}(C)$ of columns, and placing all points of $\hat{X}$ lying in C on the columns of $\mathcal{B}(C)$ so that they form a monotonically increasing sequence of length n. It is not hard to see that $\mathsf{WB}^{(2)}(X^*) \geq \mathsf{WB}^{(2)}(\hat{X})$ which concludes the proof. □

5 Guillotine bounds

In this section we define two extensions of the strong Wilber bound and extend our negative results to one of these bounds. In subSection 5.1, we provide formal definitions of these bounds, and we present our negative result in subsequent subsections.

5.1 Definitions

Assume that we are given an input set X of n points, that is a permutation. Let $\mathcal{L}^V$ be the set of all vertical lines with half-integral x-coordinates between $1/2$ and $n - 1/2$, and let $\mathcal{L}^H$ be the set of all horizontal lines with half-integral y-coordinates between $1/2$ and $n - 1/2$. Recall that for every permutation σ of $\mathcal{L}^V$, we have defined a bound $\mathsf{WB}_\sigma(X)$. We can similarly define a bound $\mathsf{WB}'_{\sigma'}(X)$ for every permutation σ' of $\mathcal{L}^H$. We also let $\mathsf{WB}'(X)$ be the maximum, over all permutations σ' of $\mathcal{L}^H$, of $\mathsf{WB}'_{\sigma'}(X)$. Equivalently, let X' be an instance obtained from X by rotating it by 90 degrees clockwise. Then $\mathsf{WB}'(X) = \mathsf{WB}(X')$. We denote by B a bounding box that contains all points of X.

5.1.1 Consistent Guillotine Bound

In this section we define the *consistent Guillotine Bound*, $\mathsf{cGB}(X)$. Let σ be any permutations of all lines in $\mathcal{L}^V \cup \mathcal{L}^H$. We start from a bounding box B containing all points of X and maintain a partition $\mathcal{P}$ of the plane into rectangular regions, where initially $\mathcal{P} = \{B\}$. We process the lines in $\mathcal{L}^V \cup \mathcal{L}^H$ according to their ordering in σ. Consider an iteration when a line L is processed. Let $P_1, \ldots, P_k$ be all rectangular regions in $\mathcal{P}$ that intersect the line L. For each such region P_j, let P'_j and P''_j be the two rectangular regions into which the line L splits P_j. We update $\mathcal{P}$ by replacing each region P_j, for $1 \leq j \leq k$, with the regions P'_j and P''_j. Once all lines in $\mathcal{L}^V \cup \mathcal{L}^H$ are processed, we terminate the process.

This recursive partitioning procedure can be naturally associated with a partitioning tree $T = T_\sigma$ that is defined as follows:

- Each vertex $v \in V(T)$ is associated with a rectangular region $S(v)$ of the plane. If r is the root of T, then $S(r) = B$.
- Each non-leaf vertex v is associated with a line $L(v) \in \mathcal{L}^H \cup \mathcal{L}^V$ that was used to partition $S(v)$ into two sub-regions, S' and S''. Vertex v has two children v_1, v_2 in T, with $S(v_1) = S'$ and $S(v_2) = S''$.
- For each leaf node v, the region $S(v)$ contains at most one point of X.

We now define the cost $\mathsf{cost}(v)$ of each node $v \in V(T)$. If the region $S(v)$ contains no points of X, or it contains a single point of X, then $\mathsf{cost}(v) = 0$. Otherwise, we define $\mathsf{cost}(v)$ in the same manner as before. Assume first that the line $L(v)$ is vertical. Let $p_1, \ldots, p_k$ be all points in $X \cap S(v)$, indexed in the increasing order of their y-coordinates. A pair (p_j, p_{j+1}) of consecutive points forms a crossing of $L(v)$ for $S(v)$, if they lie on the opposite sides of $L(v)$. We then let $\mathsf{cost}(v)$ be the number of such crossings.

When $L(v)$ is a horizontal line, $\mathsf{cost}(v)$ is defined analogously: we index the points of $X \cap S(v)$ in the increasing order of their x-coordinates. We then say that a consecutive pair of such points is a crossing iff they lie on opposite sides of $L(v)$. We let $\mathsf{cost}(v)$ be the number of such crossings.

For a fixed ordering σ of the lines in $\mathcal{L}^V \cup \mathcal{L}^H$, and the corresponding partition tree $T = T(\sigma)$, we define $\mathsf{cGB}_\sigma(X) = \sum_{v \in V(T)} \mathsf{cost}(v)$.

Lastly, we define the consistent Guillotine Bound for a point set X that is a permutation to be the maximum, over all orderings σ of the lines in $\mathcal{L}^V \cup \mathcal{L}^H$, of $\mathsf{cGB}_\sigma(X)$.

In the following subsection we define an even stronger bound, that we call the *Guillotine bound*, and we show that for every point set X that is a permutation, $\mathsf{cGB}(X) \leq \mathsf{GB}(X)$, and moreover that $\mathsf{GB}(X) \leq O(\mathsf{OPT}(X))$. It then follows that for every point set X that is a permutation, $\mathsf{cGB}(X) \leq O(\mathsf{OPT}(X))$.

Theorem 5.1. *For every integer n', there is an integer $n \geq n'$, and a set X of points that is a permutation with $|X| = n$, such that $\mathsf{OPT}(X) \geq \Omega(n \log\log n)$ but $\mathsf{cGB}(X) \leq O(n \log\log\log n)$.*

The following lemma will be helpful in the proof of Theorem 5.1; recall that $\mathsf{WB}'(X)$ is the basic Wilber Bound, where we cut via horizontal lines only.

Lemma 5.2. *For every instance X that is a permutation, $\mathsf{cGB}(X) \leq \mathsf{WB}(X) + \mathsf{WB}'(X)$.*

Proof. Let σ be a permutation of $\mathcal{L}^V \cup \mathcal{L}^H$, such that $\mathsf{cGB}(X) = \mathsf{cGB}_\sigma(X)$. Notice that σ naturally induces a permutation σ' of $\mathcal{L}^V$ and a permutation σ'' of $\mathcal{L}^H$. We show that $\mathsf{cGB}_\sigma(X) \leq \mathsf{WB}_{\sigma'}(X) + \mathsf{WB}'_{\sigma''}(X)$. In order to do so, it is enough to show that for every vertical line $L \in \mathcal{L}^V$, the cost that is charged to L in the bound $\mathsf{cGB}_\sigma(X)$ is less than or equal to the cost that is charged to L in the bound $\mathsf{WB}_{\sigma'}(X)$, and similarly, for every horizontal line $L' \in \mathcal{L}^H$, the cost that is charged to L' in the bound $\mathsf{cGB}_\sigma(X)$ is less than or equal to the cost that is charged to L' in the bound $\mathsf{WB}'_{\sigma''}(X)$. We show the former; the proof of the latter is similar.

Consider any line $L \in \mathcal{L}^V$. We let T be the partitioning tree associated with $\mathsf{cGB}_\sigma(X)$, just before line L is processed, and we let T' be defined similarly for $\mathsf{WB}_{\sigma'}(X)$. Let $v \in V(T')$ be the leaf vertex with $L \subseteq S(v)$, and let U be the set of all leaf vertices u of the tree T with $S(u) \cap L \neq \emptyset$. Observe that the set of vertical lines that appear before L in σ and σ' is identical. Therefore, $S(v) = \bigcup_{u \in U} S(u)$. It is easy to verify that, for every vertex $u \in U$, every crossing that contributes to $\mathsf{cost}(u)$ is also a crossing that is charged to the line L in the strip $S(v)$. Therefore, the total number of crossings of line L in tree T' that contribute to $\mathsf{WB}_{\sigma'}(X)$ is greater than or equal to the number of crossings of the line L that contribute to $\mathsf{cGB}_\sigma(X)$.

To conclude, we get that $\mathsf{cGB}(X) = \mathsf{cGB}_\sigma(X) \leq \mathsf{WB}_{\sigma'}(X) + \mathsf{WB}'_{\sigma''}(X) \leq \mathsf{WB}(X) + \mathsf{WB}'(X)$. $\square$

5.1.2 The Guillotine Bound

In this section, we define a second extension of Wilber Bound, that we call Guillotine Bound, and denote by GB. The bound is more convenient to define using a partitioning tree instead of a sequence of lines. Let X be a point set which is a permutation.

We define a guillotine partition of a point set X, together with the corresponding partitioning tree T. As before, every node $v \in V(T)$ of the partitioning tree T is associated with a rectangular region $S(v)$ of the plane. At the beginning, we add the root vertex r to the tree T, and we let $S(r) = B$, where B is the bounding box containing all points of X. We then iterate, as long as some leaf vertex v of T has $S(v) \cap X$ containing more than one point. In each iteration, we select any such leaf vertex v, and we select an arbitrary vertical or horizontal line $L(v)$, that is contained in $S(v)$, and partitions $S(v)$ into two rectangular regions, that we denote by S' and S'', such that $X \cap S', X \cap S'' \neq \emptyset$. We then add two child vertices v_1, v_2 to v, and set $S(v_1) = S'$ and $S(v_2) = S''$. Once every leaf vertex v has $|S(v) \cap X| = 1$, we terminate the process and obtain the final partitioning tree T.

The cost $\mathsf{cost}(v)$ of every vertex $v \in V(T)$ is calculated exactly as before. We then let $\mathsf{GB}_T(X) = \sum_{v \in V(T)} \mathsf{cost}(v)$, and we let $\mathsf{GB}(X)$ be the maximum, over all partitioning trees T, of $\mathsf{GB}_T(X)$.

We note that the main difference between $\mathsf{cGB}(X)$ and $\mathsf{GB}(X)$ is that in cGB bound, the partitioning lines must be chosen consistently across all regions: that is, we choose a vertical or a horizontal line L that crosses the entire bounding box B, and then we partition every region that intersects L by this line L. In contrast, in the GB bound, we can partition each region $S(v)$ individually, and choose different partitioning lines for different regions. It is then easy to see that GB is more general than cGB, and, in particular, for every point set X that is a permutation, $\mathsf{cGB}(X) \leq \mathsf{GB}(X)$.

Lastly, we show that GB is a lower bound on the optimal solution cost, in the following lemma.

Lemma 5.3. *For any point set X that is a permutation,* $\mathsf{GB}(X) \leq 2\mathsf{OPT}(X)$.

5.2 Negative results for the Consistent Guillotine Bound

In this section we prove Theorem 5.1.

We use three main parameters. Let $\ell \geq 1$ be an integer, and let $n = 2^\ell$ and $N = 2^n$. As before, we will first construct point set $\hat{X}$ that is not a permutation (in fact, it is not even a semi-permutation), and then we will turn it into our final instance X^* which is a permutation.

5.2.1 2D exponentially spaced bit reversal

We define the instance $\mathsf{2D\text{-}ES\text{-}BRS}(\ell)$ to be a bit-reversal sequence $\mathsf{BRS}(\ell, \mathcal{R}, C)$, where the sets $\mathcal{R}$ and C of active rows and columns are defined as follows. Set C contains all columns with x-coordinates in $\left\{2^j \mid 1 \leq j \leq n\right\}$, and similarly set $\mathcal{R}$ contains all rows with y-coordinates in $\left\{2^j \mid 1 \leq j \leq n\right\}$. Note that set X contains n points, whose x- and y-coordinates are integers between 1 and N.

5.2.2 2D cyclic shifts

Next, we define the shifted and exponentially spaced instance, but this time we shift both vertically and horizontally. We assume that we are given a horizontal shift $0 \leq s < N$ and a vertical shift $0 \leq s' < N$. In order to construct the instance $X^{s,s'}$, we start with the instance $X = \mathsf{2D\text{-}ES\text{-}BRS}(\ell)$, and then perform the following two operations. First, we perform a horizontal shift by s units as before, by moving the last s columns with integral x-coordinates to the beginning of the instance. Next, we perform a vertical shift, by moving the last s' rows with integral y-coordinates to the bottom of the instance. We let $X^{s,s'}$ denote the resulting instance. By applying Claim 4.3 twice, once for the horizontal shift, and once for the vertical shift, we get that $\mathsf{OPT}(X^{s,s'}) \geq \mathsf{OPT}(X) - 2|X| \geq \Omega(\log N \log\log N)$, since $|X| = \log N$.

5.2.3 Instance $\hat{X}$

Next, we construct an instance $\hat{X}$, by combining the instances $X^{s,s'}$ for $0 \leq s, s' < N$. In order to do so, let $\hat{C}$ be a set of N^2 columns, with integral x-coordinates $1, \ldots, N^2$. We partition $\hat{C}$ into subsets $C_1, C_2, \ldots, C_N$, each of which contains N consecutive columns, they appear in this left-to-right order. We call each such set C_i a *super-column*. We denote by S^V_i the smallest vertical strip containing all columns of C_i.

Similarly, we let $\hat{\mathcal{R}}$ be a set of N^2 rows, with integral y-coordinates $1, \ldots, N^2$. We partition $\hat{\mathcal{R}}$ into subsets $\mathcal{R}_1, \ldots, \mathcal{R}_N$, each of which contains N consecutive rows, such that $\mathcal{R}_1, \ldots, \mathcal{R}_N$ appear in this bottom-to-top order. We call each such set $\mathcal{R}_i$ a *super-row*. We denote by S^H_i the smallest horizontal strip containing all rows of $\mathcal{R}_i$. For all $1 \leq i, j \leq N$, we let $B(i, j)$ be the intersection of the horizontal strip S^H_i and the vertical strip S^V_i. We plant the instance $X^{(i-1),(j-1)}$

into the box $B(i, j)$. This completes the construction of instance $\hat{X}$. Let $N^* = |\hat{X}| = N^2 \log N$ (recall that each instance $X^{s,s'}$ contains $\log N$ points.)

Observe that, for each vertical strip S_i^V, all instances planted into S_i^V have the same vertical shift - $(i-1)$; the horizontal shift s' of each instance increases from 0 to $N-1$ as we traverse S_i^V from bottom to top. In particular, the instance planted into S_1^V is precisely the instance $\hat{X}$ from Section 4.2 (if we ignore inactive rows). For each $i > 1$, the instance planted into S_i^V is very similar to the instance $\hat{X}$ from Section 4.2, except that each of its corresponding sub-instances is shifted vertically by exactly $(i-1)$ rows.

Similarly, for each horizontal strip S_j^H, all instances planted into S_j^H have the same horizontal shift - $(j-1)$; the vertical shift s' of each instance increases from 0 to $N-1$ as we traverse S_j^H from left to right.

Since, for every instance $X^{s,s'}$, $\mathsf{OPT}(X^{s,s'}) = \Omega(\log N \log\log N)$, we obtain the following bound.

Observation 5.4. $\mathsf{OPT}(\hat{X}) = \Omega(N^2 \log N \log\log N) = \Omega(N^* \log\log N^*)$.

Since instance $\hat{X}$ is symmetric, and every point lies on one of the N^2 rows of $\hat{\mathcal{R}}$ and on one of the N^2 rows of $\hat{\mathcal{C}}$, we obtain the following.

Observation 5.5. Every row in $\hat{\mathcal{R}}$ contains exactly $\log N$ points of $\hat{X}$. Similarly, every column of $\hat{\mathcal{C}}$ contains exactly $\log N$ points of $\hat{X}$.

5.2.4 Final instance

Lastly, in order to turn $\hat{X}$ into a permutation X^*, we perform a similar transformation to that in Section 4.2: for every column $C \in \mathcal{C}$, we replace C with a collection $\mathcal{B}(C)$ of $\log N$ consecutive columns, and we place all points that lie on C on the columns of $\mathcal{B}(C)$, so that they form an increasing sequence, while preserving their y-coordinates. We replace every row $R \in \mathcal{R}$ by a collection $\mathcal{B}(R)$ of $\log N$ rows similarly. The resulting final instance X^* is now guaranteed to be a permutation, and it contains $N^* = N^2 \log N$ points. Using the same reasoning as in Section 4.2, it is easy to verify that $\mathsf{OPT}(X^*) \geq \mathsf{OPT}(\hat{X}) \geq \Omega(N^* \log\log N^*)$. In the remainder of this section, we will show that $\mathsf{cGB}(X^*) = O(N^* \log\log\log N^*)$.

Abusing the notation, for all $1 \leq i \leq N^2$, we denote by S_i^V the vertical strip obtained by taking the union of all blocks $\mathcal{B}(C)$ of columns, where C belonged to the original strip S_i^V. We define the horizontal strips S_i^H similarly. Note that, from Lemma 5.2, it is enough to prove that $\mathsf{WB}(X^*) = O(N^* \log\log\log N^*)$ and that $\mathsf{WB}'(X^*) = O(N^* \log\log\log N^*)$. We do so in the following two subsections.

5.3 Handling vertical cuts

The goal of this section is to prove the following theorem:

Theorem 5.6. $\mathsf{WB}(X^*) \leq O(N^* \log\log\log N^*)$.

For all $1 \leq i \leq N$, we denote by $\mathcal{B}_i$ the set of active columns that lie in the vertical strip S_i^V, so that $\mathcal{B}_1, \ldots, \mathcal{B}_N$ partition the set of active columns of X^*. Let $\mathcal{L}'$ be a collection of lines at half-integral coordinates that partitions the bounding box B into N strips where each strip contains exactly the block $\mathcal{B}_i$ of columns. We consider the split of X^* by the lines $\mathcal{L}'$: This is a collection of N strip instances (that we will denote by $X_1^*, \ldots, X_N^*$) and a compressed instance, that we denote by $\tilde{X}$. In order to prove Theorem 5.6, we bound $\mathsf{WB}(X_i^*)$ for every strip instance X_i^*, and $\mathsf{WB}(\tilde{X})$ for the compressed instance $\tilde{X}$, and then combine them using Theorem 3.6 in order to obtain the final bound on $\mathsf{WB}(X^*)$.

5.3.1 Bounding Wilber bound for strip instances

In this subsection, we prove the following lemma.

Lemma 5.7. *For all* $1 \leq i \leq N$, $\mathsf{WB}(X_i^*) \leq O(N \log N \log\log\log N)$.

From now on we fix an index i, and consider the instance X_i^*. Recall that in order to construct instance X_i^*, we started with the instances $X^{0,i}, X^{1,i}, \ldots, X^{N-1,i}$, each of which has the same vertical shift (shift i), and horizontal shifts ranging from 0 to $N-1$. Let $\hat{X}_i$ be the instance obtained by stacking these instances one on top of the other, similarly to the construction of instance $\hat{X}$ in Section 4.2. As before, instance $\hat{X}_i$ is a semi-permutation, so every row contains at most one point. Every column of $\hat{X}_i$ contains exactly $\log N$ points of $\hat{X}_i$. Let $\mathcal{C}$ denote the set of all active columns of instance $\hat{X}_i$. For every column $C \in \mathcal{C}$, we replace C with a block $\mathcal{B}(C)$ of $\log N$ columns, and place all points of $\hat{X}_i \cap C$ on the columns of $\mathcal{B}(C)$, so that they form an increasing sequence, while preserving their y-coordinates. The resulting instance is equivalent to X_i^* (to obtain instance X_i^* we also need to replace every active row R with a block $\mathcal{B}(R)$ of $\log N$ rows; but since every row contains at most one point of $\hat{X}_i$, this amounts to inserting empty rows into the instance).

The analysis of $\mathsf{WB}(X_i^*)$ is very similar to the analysis of $\mathsf{WB}(X^*)$ for instance X^* constructed in Section 4.2. Notice that, as before, it is sufficient to show that $\mathsf{WB}(\hat{X}_i) \leq O(N \log N \log\log\log N)$. Indeed, consider the partition $\{\mathcal{B}(C)\}_{C \in \mathcal{C}}$ of the columns of X_i^*. Then $\hat{X}_i$ can be viewed as the compressed instance for X_i^* with the respect to this partition. Each resulting strip instance (defined by the block $\mathcal{B}(C)$ of columns) is an increasing sequence of $\log N$ points, so the Wilber Bound value for such an instance is $O(\log N)$. Altogether, the total Wilber Bound of all such strip instances is $O(N \log N)$. Therefore, from Theorem 3.6, in order to prove Lemma 5.7, it is now sufficient to show that $\mathsf{WB}(\hat{X}_i) \leq O(N \log N \log\log\log N)$.

Let $\mathcal{L}$ be the set of all vertical lines with half-integral coordinates for the instance $\hat{X}_i$, and let σ be any permutation of these lines. Our goal is to prove that $\mathsf{WB}_\sigma(\hat{X}_i) \leq O(N \log N \log\log\log N)$. Let $T = T(\sigma)$ be the partitioning tree associated with σ. Consider some vertex $v \in V(T)$ and the line L that v owns. As before, we classify crossings that are charged to L into several types. A crossing (p, p') is a type-1 crossing, if p and p' both lie in the same instance $X^{j,i}$. We say that instance $X^{j,i}$ is *bad* for L, if it contributes at least one type-1 crossing to the cost of L. If $p \in X^{j,i}$ and $p' \in X^{j',i}$ for $j \neq j'$, then we say that (p, p') is a type-2 crossing. If either instance $X^{j,i}$ or $X^{j',i}$ is a bad instance for L, then the crossing is of type (2a); otherwise it is of type (2b).

We now bound the total number of crossings of each of these types separately.

- **Type-1 Crossings.** We bound the total number of all type-1 crossings exactly like in Section 4.3. We note that the proof does not use the vertical locations of the points in the sub-instances $X^{j,i}$, and only relies on two properties of instance $\hat{X}$: (i) the points in the first instance X_0 (corresponding to instance $X^{0,i}$) are exponentially spaced horizontally, so the x-coordinates of the points are integral powers of 2, and they are all distinct; and (ii) each subsequent instance X_s (corresponding to instance $X^{s,i}$) is a copy of X_0 that is shifted horizontally by s units. Therefore, the same analysis applies, and the total number of type-1 crossings in $\hat{X}_i$ can be bounded by $O(N \log N \log\log\log N)$ as before.

- **Type-(2a) Crossings.** As before, we charge each type-(2a) crossing to one of the corresponding bad instances, to conclude that the total number of type-(2a) crossings is bounded by the total number of type-1 crossings, which is in turn bounded by $O(N \log N \log\log\log N)$.

- **Type-(2b) Crossings.** Recall that in order to bound the number of type-(2b) crossings, we have collapsed, for every instance X_s, all rows of X_s into a single row. If we similarly collapse, for every instance $X^{s,i}$, all rows of this instance into a single row, we will obtain an identical set of points. This is because the only difference between instances X_s and $X^{s,i}$ is vertical position of their points. Therefore, the total number of type-(2b) crossings in $\hat{X}_i$ is bounded by $O(N \log N)$ as before.

  This finishes the proof of Lemma 5.7. We conclude that

$$\sum_{i=1}^{N} \mathsf{WB}(X_i^*) \leq O(N^2 \log N \log\log\log N) = O(N^* \log\log\log N^*) . \tag{5.1}$$

5.3.2 Bounding Wilber bound for the compressed instance

In this subsection, we prove the following lemma.

Lemma 5.8. $\mathsf{WB}(\tilde{X}) \leq O(N^*)$.

We denote the active columns of $\tilde{X}$ by $C_1, \ldots, C_N$. Recall that each column C_i contains exactly $N \log N$ input points. Let $\mathcal{R}$ be the set of all rows with integral coordinates, so $|\mathcal{R}| = N^2 \log N$. Let $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_{N^2}$ be a partition of the rows in $\mathcal{R}$ into blocks containing $\log N$ consecutive rows each, where the blocks are indexed in their natural bottom-to-top order. Recall that each such block $\mathcal{B}_i$ represents some active row of instance $\hat{X}$, and the points of $\tilde{X}$ that lie on the rows of $\mathcal{B}_i$ form an increasing sequence. We also partition the rows of $\mathcal{R}$ into super-blocks, $\hat{\mathcal{B}}_1, \ldots, \hat{\mathcal{B}}_N$, where each superblock is the union of exactly N consecutive blocks. For each subinstance $X^{s,s'}$, the points of the subinstance lie on rows that belong to a single super-block.

Let $\mathcal{L}$ be the set of all columns with half-integral coordinates for $\tilde{X}$, so $|\mathcal{L}| \leq N$. We fix any permutation σ of $\mathcal{L}$, and prove that $\mathsf{WB}_\sigma(\tilde{X}) \leq O(N^*)$. Let T be the partitioning tree associated with the permutation σ.

Consider any vertex $v \in V(T)$, its corresponding vertical strip $S = S(v)$, and the vertical line $L = L(v)$ that v owns. Let (p, p') be a crossing of L, so p and p' both lie in S on opposite sides of L, and no point of $\tilde{X} \cap S$ lies between the row of p and the row of p'. Assume that the row of p is below the row of p'. We say that the crossing is *left-to-right* if p is to the left of L, and we say that it is *right-to-left* otherwise. In order to bound the number of crossings, we use the following two claims.

Claim 5.9. *Assume that (p, p') is a left-to-right crossing, and assume that p lies on a row of $\mathcal{B}_i$ and p' lies on a row of $\mathcal{B}_j$, with $i \leq j$. Then either $j \leq i+1$ (so the two blocks are either identical or consecutive), or block $\mathcal{B}_i$ is the last block in its super-block.*

Proof. Assume that p lies on column $C_{s'}$ and on a row of super-block $\hat{\mathcal{B}}_s$, so this point originally belonged to instance $X^{s,s'}$. Recall that instance $X^{s,s'+1}$ (that lies immediately to the right of $X^{s,s'}$) is obtained by circularly shifting all points in instance $X^{s,s'}$ by one unit up. In particular, a copy p^c of p in $X^{s,s'+1}$ should lie one row above the copy of p in $X^{s,s'}$, unless p lies on the last row of $X^{s,s'}$. In the latter case, block $\mathcal{B}_i$ must be the last block of its superblock $\hat{\mathcal{B}}_s$. In the former case, since point p^c does not lie between the row of p and the row of p', and it lies on column C_{s+1}, the block of rows in which point p' lies must be either $\mathcal{B}_i$ or $\mathcal{B}_{i+1}$, that is, $j \leq i+1$. □

Claim 5.10. *Assume that (p, p') is a right-to-left crossing, and assume that p lies on a row of $\mathcal{B}_i$ and p' lies on a row of $\mathcal{B}_j$, with $i \leq j$. Then either (i) $j \leq i+1$ (so the two blocks are identical or consecutive); or (ii) block $\mathcal{B}_i$ is the last block in its super-block; or (iii) block $\mathcal{B}_j$ is the first block it its super-block; or (iv) p lies on the last active column in strip S, and p' lies on the first active column in strip S.*

Proof. Assume that p lies on column $C_{s'}$ and on a row of superblock $\hat{\mathcal{B}}_s$, so this point originally belonged to instance $X^{s,s'}$. Assume for that $C_{s'}$ is not the last active column of S, so $C_{s'+1}$ also lies in S.

Recall that instance $X^{s,s'+1}$ is obtained by circularly shifting all points in instance $X^{s,s'}$ by one unit up. In particular, a copy p^c of p in $X^{s,s'+1}$ should lie one row above the copy of p in $X^{s,s'}$, unless p lies on the last row of $X^{s,s'}$. In the latter case, block $\mathcal{B}_i$ must be the last block of its superblock. In the former case, since point p^c does not lie between the row of p and the row of p', the block of rows in which point p' lies is either $\mathcal{B}_i$ or $\mathcal{B}_{i+1}$, that is, $j \leq i+1$.

Using a symmetric argument, if p' does not lie on the first active column of S, then either $j \leq i+1$, or $\mathcal{B}_j$ is the first block in its super-block. □

We can now categorize all crossings charged to the line L into types as follows. Let (p, p') be a crossing, and assume that p lies on a row of $\mathcal{B}_i$, p' lies on a row of $\mathcal{B}_j$, and $i \le j$. We say that (p, p') is a crossing of type 1, if $j \le i+1$. We say that it is a crossing of type 2 if either $\mathcal{B}_i$ or $\mathcal{B}_j$ are the first or the last blocks in their superblock. We say that it is of type 3 if p lies on the last active column of S and p' lies on the first active column of S.

We now bound the total number of all such crossings separately.

- **Type-1 crossings** Consider any pair $\mathcal{B}_i, \mathcal{B}_{i+1}$ of consecutive blocks, and let $\tilde{X}'_i$ be the set of all points lying on the rows of these blocks. Recall that all points lying on the rows of $\mathcal{B}_i$ form an increasing sequence of length $\log N$, and the same is true for all points lying on the rows of $\mathcal{B}_{i+1}$. It is then easy to see that $\mathsf{OPT}(\tilde{X}'_i) \le O(\log N)$, and so the total contribution of crossings between the points of $\tilde{X}'_i$ to $\mathsf{WB}_\sigma(\tilde{X})$ is bounded by $O(\log N)$. Since the total number of blocks $\mathcal{B}_i$ is bounded by N^2, the total number of type-1 crossings is at most $O(N^2 \log N)$.

- **Type-2 crossings** In order to bound the number of type-2 crossings, observe that $|\mathcal{L}| \le N$. If $L \in \mathcal{L}$ is a vertical line, and S is a strip that L splits, then there are N superblocks of rows that can contribute type-2 crossings to $\mathsf{cost}(L)$, and each such superblock may contribute at most one crossing. Therefore, the total number of type-2 crossings charged to L is at most N, and the total number of all type-2 crossings is $O(N^2)$.

- **Type-3 crossings** In order to bound the number of type-3 crossings, observe that every column contains $N \log N$ points. Therefore, if $L \in \mathcal{L}$ is a vertical line, then the number of type-3 crossings charged to it is at most $2N \log N$. As $|\mathcal{L}| \le N$, we get that the total number of type-3 crossings is $O(N^2 \log N)$.

To conclude, we have shown that $\mathsf{WB}_\sigma(\tilde{X}) = O(N^2 \log N) = O(N^*)$, proving Lemma 5.8. By combining Lemmas 5.7 and 5.8, together with Theorem 3.6, we conclude that $\mathsf{WB}(X^*) = O(N^* \log\log\log N^*)$, proving Theorem 5.6.

5.4 Handling horizontal cuts

We show the following analogue of Theorem 5.6.

Theorem 5.11. *$\mathsf{WB}'(X^*) \le O(N^* \log\log\log N^*)$.*

The proof of the theorem is virtually identical to the proof of Theorem 5.6. In fact, consider the instance X^{**}, that is obtained from X^*, by rotating it by 90 degrees clockwise. Consider the sequence $\mathsf{BRS}'(\ell, \mathcal{R}, C)$ that is obtained by rotating the point set $\mathsf{BRS}(\ell, \mathcal{R}, C)$ by 90 degrees. Consider now the following process. Our starting point is the rotated Bit Reversal Sequence. We then follow exactly the same steps as in the construction of the instance X^*. Then the resulting instance is precisely (a mirror reflection of) instance X^{**}. Notice that the only place where our proof uses the fact that we start with the Bit Reversal Sequence is in order to show that $\mathsf{OPT}(X^*)$ is sufficiently large. In fact we could replace the Bit Reversal Sequence with any other point set that is a permutation, and whose optimal solution cost is as large, and in particular the Bit Reversal Sequence that is rotated by 90 degrees would work just as well. The analysis of the Wilber Bound works exactly as before, and Theorem 5.11 follows.

6 Algorithmic results

6.1 Overview

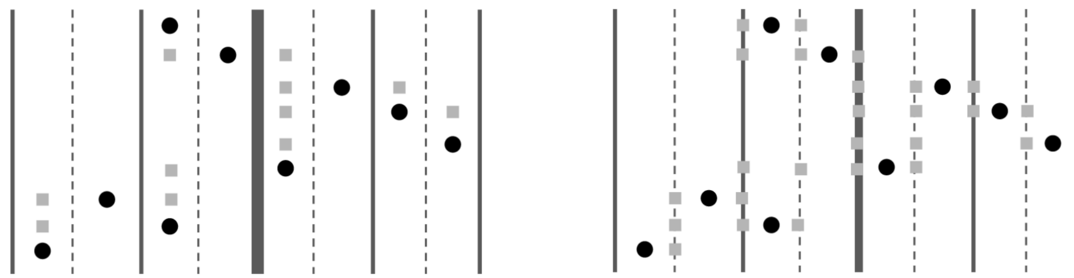
We provide the high level intuition for the proof of Theorem 1.2. Both the polynomial time and the subexponential time algorithms follow the same framework. We start with a high-level overview of this framework. For simplicity, assume that the number of active columns in the input instance X is an integral power of 2. The key idea is to decompose the input instance into smaller sub-instances, using the split instances defined in Section 3.1. We solve the resulting instances recursively and then combine the resulting solutions.

Suppose we are given an input point set X that is a semi-permutation, with $|X| = m$, such that the number of active columns is n. We consider a *balanced* partitioning tree T, where for every vertex $v \in V(T)$, the line $L(v)$ that v owns splits the strip $S(v)$ in the middle, with respect to the active columns that are contained in $S(v)$. Therefore, the height of the partitioning tree is $\log n$.

Consider now the set U of vertices of T that lie in the middle layer of T. Let $\mathcal{L}'$ be their strip boundaries, so we have $|\mathcal{L}'| = \Theta(\sqrt{n})$. Consider the split of X by $\mathcal{L}'$, obtaining a new collection of instances $(\widetilde{X}, \{X_i\})_{i=1}^k$ where $k = \Theta(\sqrt{n})$. Note that each resulting strip instance X_i contains $\Theta(\sqrt{n})$ active columns, and so does the compressed instance $\widetilde{X}$.

We recursively solve each such instance and then combine the resulting solutions. The key to the algorithm and its analysis is to show that there is a collection Z of $O(|X|)$ points, such that, if we are given any solution $\widehat{Y}$ to instance $\widetilde{X}$, and, for all $1 \le i \le k$, any solution Y_i to instance X_i, then $Z \cup \widehat{Y} \cup \left(\bigcup_{i=1}^k Y_i\right)$ is a feasible solution to instance X. We also show that the total number of input points that appear in all instances that participate in the same recursive level is bounded by $O(\mathsf{OPT}(X))$. This ensures that in every recursive level we add at most $O(\mathsf{OPT}(X))$ points to the solution, and the total solution cost is at most $O(\mathsf{OPT}(X))$ times the number of the recursive levels, which is bounded by $O(\log \log n)$.

In order to obtain the subexponential time algorithm, we restrict the recursion to D levels, and then solve each resulting instance X' directly in time $r(X')c(X')^{O(c(X'))}$. This approach gives an $O(D)$-approximation algorithm with running time at most $\mathrm{poly}(m) \cdot \exp\left(n^{1/2^{\Omega(D)}} \log n\right)$ as desired.



(a) Canonical Solution (b) Special Solution

Figure 7: Canonical and T-special solutions of X. The input points are shown as circles; the points that belong to the solution Y are shown as squares.

6.2 Special solutions and reduced sets

Our algorithm will produce feasible solutions of a special form, that we call *special solutions*. Recall that, given a semi-permutation point set X, the auxiliary columns for X are a set $\mathcal{L}$ of vertical lines with half-integral coordinates. We say that a solution Y for X is *special* iff every point of Y lies on an row that is active for X, and on a column of $\mathcal{L}$. In particular, special solutions are by definition non-canonical (see Figure 7 for an illustration).

If σ' is any ordering of the auxiliary columns in $\mathcal{L}' \subseteq \mathcal{L}$, and $T' = T(\sigma')$ is the corresponding partitioning tree, then any point set Y that is a special solution for X is also called a *T'-special solution*: The solutions use only auxiliary columns in $\mathcal{L}'$ (equivalently, strip boundaries of $S(v)$ for $v \in V(T')$).

Consider a semi-permutation X, that we think of as a potential input to the Min-Sat problem. We denote $X = \{p_1, \ldots, p_m\}$, where the points are indexed in their natural bottom-to-top order, so $(p_1).y < (p_2).y < \ldots < (p_m).y$. A point p_i is said to be *redundant*, if and only if the points immediately above and below are on its column, that is, for some i, $(p_i).x = (p_{i+1}).x = (p_{i-1}).x$. We say that a semi-permutation X is in the *reduced form* if there are no redundant points in X. The following lemma relates the optimal solutions of any instance and its reduced form.

Lemma 6.1. *Let X be a semi-permutation, and let $X' \subseteq X$ be any point set, that is obtained from X by repeatedly removing redundant points. Then* $\mathsf{OPT}(X') \le \mathsf{OPT}(X)$. *Moreover, if Y is a feasible solution for X' such that every point of Y lies on a row that is active for X', then Y is also a feasible solution for X.*

Proof. For the first claim, it is sufficient to show that, if X'' is a set of points obtained from X by deleting a single redundant point p_i, then $\mathsf{OPT}(X'') \le \mathsf{OPT}(X)$. Let R denote the row on which point p_i lies, and let R' be the row containing p_{i-1}. Let Y be the optimal solution to instance X. We assume w.l.o.g. that Y is a canonical solution. Consider the set $Z = X \cup Y$ of point, and let Z' be obtained from Z by collapsing the rows R, R' into the row R' (since Y is a canonical solution for X, no points of $X \cup Y$ lie strictly between the rows R, R'). From Observation 2.4, Z'

remains a satisfied point set. Setting $Y' = Z' \setminus X''$, it is easy to verify that Y' is a feasible solution to instance X'', and moreover, $|Y'| \leq |Y|$. Therefore, $\mathsf{OPT}(X'') \leq |Y'| \leq |Y| \leq \mathsf{OPT}(X)$.

As for the second claim, it is sufficient to show that, if X'' is a set of points obtained from X by deleting a single redundant point p_i, and Y is any canonical solution for X'', then Y is also a feasible solution for X. We can then apply this argument iteratively, until we obtain a set of points that is in a reduced form.

Consider any feasible canonical solution Y to instance X''. We claim that $X \cup Y$ is a feasible set of points. Indeed, consider any two points $p, q \in X \cup Y$ that are not aligned. If both points are distinct from the point p_i, then they must be satisfied in $X \cup Y$, since both these points lie in $X'' \cup Y$. Therefore, we can assume that $p = p_i$. Notice that $q \neq p_{i-1}$ and $q \neq p_{i+1}$, since otherwise p and q must be aligned. Moreover, q cannot lie strictly between the row of p_{i-1} and the row of p_{i+1}, as we have assumed that every point of Y lies on a row that is active for X' X''. But then it is easy to verify that either point p_{i-1} lies in $\square_{p,q}$ (if q is below p), or point p_{i+1} lies in $\square_{p,q}$ (otherwise). In either case, the pair (p, q) is satisfied in $X \cup Y$. $\square$

From Lemma 6.1, whenever we need to solve the Min-Sat problem on an instance X, it is sufficient to solve it on a sub-instance, obtained by iteratively removing redundant points from X. We obtain the following immediate corollary of Lemma 6.1.

Corollary 6.2. *Let X be a semi-permutation, and let $X' \subseteq X$ be any point set, that is obtained from X by repeatedly removing redundant points. Let Y be any special feasible solution for X'. Then Y is also a special feasible solution for X.*

Lastly, we need the following lemma, which is a simple application of the Wilber bound.

Lemma 6.3. *Let X be a point set that is a semi-permutation in reduced form. Then* $\mathsf{OPT}(X) \geq |X|/4 - 1$.

Proof. Since X is a semi-permutation, every point of X lies on a distinct row; we denote $|X| = n$. Let $X = \{p_1, \ldots, p_n\}$, where the points are indexed in the increasing order of their y-coordinates. Let $\Pi = \{(p_i, p_{i+1}) \mid 1 \leq i < n\}$ be the collection of all consecutive pairs of points in X. We say that the pair (p_i, p_{i+1}) is *bad* iff both p_i and p_{i+1} lie on the same column. From the definition of the reduced form, if (p_i, p_{i+1}) is a bad pair, then both (p_{i-1}, p_i) and (p_{i+1}, p_{i+2}) are good pairs. Let $\Pi' \subseteq \Pi$ be the subset containing all good pairs. Then $|\Pi'| \geq (|\Pi| - 1)/2 \geq n/2 - 1$. Next, we select a subset $\Pi'' \subseteq \Pi'$ of pairs, such that $|\Pi''| \geq |\Pi'|/2 \geq n/4 - 1$, and every point in X belongs to at most one pair in Π''. Since every point in X belongs to at most two pairs in Π', it is easy to see that such a set exists. Let Y be an optimal solution to instance X.

Consider now any pair (p_i, p_{i+1}) of points in Π''. Then there must be a point $y_i \in Y$ that lies in the rectangle $\square_{p_i, p_{i+1}}$. Moreover, since all points of X lie on distinct rows, and each such point belongs to at most one pair in Π'', for $i \neq j$, $y_i \neq y_j$. Therefore, $|Y| \geq |\Pi''| \geq n/4 - 1$. $\square$

6.3 Our algorithm

Suppose we are given an input set X of points that is a semi-permutation. Let T be any partitioning tree for X. We say that T is a *balanced* partitioning tree for X iff for every non-leaf vertex $v \in V(T)$ and its children v_1, v_2, the number of active columns inside $S(v_1)$ and $S(v_2)$ are roughly the same, that is, $c(X \cap S(v_i)) \leq \lceil c(X \cap S(v))/2 \rceil$ for each $i = 1, 2$.

Given a partitioning tree T, we denote by Λ_i the set of all vertices of T that lie in the ith *layer* of T – that is, the vertices whose distance from the root of T is i (so the root belongs to Λ_0). The *height* of the tree T, denoted by height(T), is the largest index i such that $\Lambda_i \neq \emptyset$. If the height of the tree T is h, then we call the set $\Lambda_{\lceil h/2 \rceil}$ of vertices the *middle layer* of T. Notice that, if T is a balanced partitioning tree for input X, then its height is at most $2 \log c(X)$. The strips $\{S(v)\}_{v \in \Lambda_{\lceil h/2 \rceil}}$ are called the *middle-layer strips*.

Our algorithm takes as input a set X of points that is a semi-permutation, a balanced partition tree T for X, and an integral parameter $\rho > 0$.

Intuitively, the algorithm uses the splitting operation to partition the instance X into subinstances that are then solved recursively, until it obtains a collection of instances whose corresponding partitioning trees have height at most ρ. We then employ dynamic programming. The algorithm returns a special feasible solution for the instance. Recall that the height of the tree T is bounded by $2 \log c(X) \leq 2 \log n$. The following theorem (whose proof appears in Section 6.6) is used as a recursion basis.

Theorem 6.4. *There is an algorithm called* LEAFBST *that, given a semi-permutation instance X of* Min-Sat *in reduced form, and a partitioning tree T for it, produces a feasible T-special solution for X of cost at most* $2|X| + 2\mathsf{OPT}(X)$*, in time* $|X|^{O(1)} \cdot c(X)^{O(c(X))}$.

We now provide a schematic description of our algorithm.

RECURSIVEBST**(**X, T, ρ**)**

1. Keep removing redundant points from X until X is in reduced form.
2. If T has height at most ρ,
3. **return** LEAFBST(X, T)
4. Let $\mathcal{L}' \subseteq \mathcal{L}$ be the strip boundaries of the middle-layer strips $\{S_1, \ldots, S_k\}$ of T.
5. Compute the split $(\widetilde{X}, \{X_j\}_{j \in [k]})$ of X by $\mathcal{L}'$.
6. Compute the corresponding split subtrees $(\widetilde{T}, \{T_j\}_{j \in [k]})$ of T by $\mathcal{L}'$.
7. For $j \in [k]$, call to RECURSIVEBST with input (X_j, T_j, ρ), and let Y_j be the solution returned by it.
8. Call RECURSIVEBST with input $(\widetilde{X}, \widetilde{T}, \rho)$, and let $\hat{Y}$ be the solution returned by it.
9. Let Z be a point set containing, for each $j \in [k]$, for each point $p \in X_j$, two copies p' and p'' of p with $p'.y = p''.y = p.y$, where p' lies on the left boundary of S_j, and p'' lies on the right boundary of S_j.
10. **return** $Y^* = Z \cup \hat{Y} \cup (\bigcup_{j \in [k]} Y_j)$

The cost and feasibility analyses appear in the next two subsections.

6.4 Cost analysis

In order to analyze the solution cost, consider the final solution Y^* to the input instance X. We distinguish between two types of points in Y^*: a point $p \in Y^*$ is said to be of type 2 if it was added to the solution by Algorithm LEAFBST, and otherwise we say that it is of type 1. We start by bounding the number of points of type 1 in Y^*.

Claim 6.5. *The number of points of type 1 in the solution Y^* to the original instance X is at most $O(\log(\text{height}(T)/\rho)) \cdot \mathsf{OPT}(X)$.*

Proof. Observe that the number of recursive levels is bounded by $\lambda = O(\log(\text{height}(T)/\rho))$. This is since, in every recursive level, the heights of all trees decrease by a constant factor, and we terminate the algorithm once the tree heights are bounded by ρ. For each $1 \leq i \leq \lambda$, let $\mathcal{X}_i$ be the collection of all instances in the ith recursive level, where the instances are in the reduced form. Notice that the only points that are added to the solution by Algorithm RECURSIVEBST directly are the points in the sets Z. The number of such points added at recursive level i is bounded by

$\sum_{X' \in \mathcal{X}_i} 2|X'|$. It is now sufficient to show that for all $1 \le i \le \lambda$, $\sum_{X' \in \mathcal{X}_i} |X'| \le O(\mathsf{OPT}(X))$. We do so using the following observation.

Observation 6.6. For all $1 \le i \le \lambda$, $\sum_{X' \in \mathcal{X}_i} \mathsf{OPT}(X') \le \mathsf{OPT}(X)$.

Assume first that the observation is correct. From Lemma 6.3, $|X'| \le O(\mathsf{OPT}(X'))$. Therefore, the number of type-1 points added to the solution at recursive level i is bounded by $O(\mathsf{OPT}(X))$. We now turn to prove Observation 6.6.

Proof of Observation 6.6. The proof is by induction on the recursive level i. It is easy to see that the claim holds for $i = 1$, since, from Lemma 6.1, removing redundant points from X to turn it into reduced form cannot increase $\mathsf{OPT}(X)$.

Assume now that the claim holds for level $i-1$, and consider some level-i instance $X' \in \mathcal{X}_i$. Let $(\widetilde{X}, \{X_j\}_{j \in [k]})$ be the split of X' that we computed. Then, from Theorem 3.3, $\sum_{j \in [k]} \mathsf{OPT}(X_j) + \mathsf{OPT}(\widetilde{X}) \le \mathsf{OPT}(X')$. Since, from Lemma 6.1, removing redundant points from an instance does not increase its optimal solution cost, the observation follows. □

□

In order to obtain an efficient $O(\log\log n)$-approximation algorithm, we set ρ to be a constant (it can even be set to 1), and we use algorithm LEAFBST whenever the algorithm calls to subroutine LEAFBST. Observe that the depth of the recursion is now bounded by $O(\log\log n)$, and so the total number of type-1 points in the solution is bounded by $O(\log\log n) \cdot \mathsf{OPT}(X)$. Let $\mathcal{I}$ denote the set of all instances to which Algorithm LEAFBST is applied. Using the same arguments as in Claim 6.5, $\sum_{X' \in \mathcal{I}} |X'| = O(\mathsf{OPT}(X))$. The number of type-2 points that Algorithm LEAFBST adds to the solution for each instance $X' \in \mathcal{I}$ is bounded by $O(\mathsf{OPT}(X') + |X'|) = O(|X'|)$. Therefore, the total number of type-2 points in the solution is bounded by $O(\mathsf{OPT}(X))$. Overall, we obtain a solution of cost at most $O(\log\log n) \cdot \mathsf{OPT}(X)$, and the running time of the algorithm is polynomial in $|X|$.

Finally, in order to obtain the subexponential time algorithm, we set the parameter ρ to be such that the recursion depth is bounded by D. Since the number of active columns in instance X is $c(X)$, and the height of the partitioning tree T is bounded by $2\log c(X)$, while the depth of the recursion is at most $2\log(\text{height}(T)/\rho)$, it is easy to verify that $\rho = O\left(\frac{\log c(X)}{2^{D/2}}\right) = \frac{\log c(X)}{2^{\Omega(D)}}$. As before, let $\mathcal{I}$ be the set of all instances to which Algorithm LEAFBST is applied. Using the same arguments as in Claim 6.5, $\sum_{X' \in \mathcal{I}} (|X'| + \mathsf{OPT}(X')) = O(\mathsf{OPT}(X))$. For each such instance X', Algorithm LEAFBST produces a solution of cost $O(|X'| + \mathsf{OPT}(X'))$. Therefore, the total number of type-2 points in the final solution is bounded by $O(\mathsf{OPT}(X))$. The total number of type-1 points in the solution is therefore bounded by $O(D) \cdot \mathsf{OPT}(X)$ as before. Therefore, the algorithm produces a factor-$O(D)$-approximate solution. Finally, in order to analyze the running time of the algorithm, we first bound the running time of all calls to procedure LEAFBST. The number of such calls is bounded by $|X|$. Consider now some instance

$X' \in \mathcal{I}$, and its corresponding partitioning tree T'. Since the height of T' is bounded by ρ, we get that $c(X') \le 2^{\rho} \le 2^{\log c(X)/2^{\Omega(D)}} \le (c(X))^{1/2^{\Omega(D)}}$. Therefore, the running time of LeafBST on instance X' is bounded by $|X'|^{O(1)} \cdot (c(X'))^{O(c(X'))} \le |X'|^{O(1)} \cdot \exp\left(O(c(X')\log c(X')\right) \le |X'|^{O(1)} \cdot \exp\left(c(X)^{1/2^{\Omega(D)}} \cdot \log c(X)\right)$.

The running time of the remainder of the algorithm, excluding the calls to LeafBST, is bounded by poly($|X|$). We conclude that the total running time of the algorithm is bounded by

$$|X|^{O(1)} \cdot \exp\left(c(X)^{1/2^{\Omega(D)}} \cdot \log c(X)\right) \le \mathrm{poly}(m) \cdot \exp\left(n^{1/2^{\Omega(D)}} \cdot \log n\right)$$

6.5 Feasibility

We start by showing that the solution that the algorithm returns is T-special.

Observation 6.7. Assuming that LeafBST(X, T) returns a T-special solution, the solution Y^* returned by Algorithm RecursiveBST(X, T, ρ) is a T-special solution.

Proof. The proof is by induction on the recursion depth. The base of the induction is the calls to Procedure LeafBST(X, T), which return T-special solutions by our assumption. Consider now some call to Algorithm RecursiveBST(X, T, ρ). From the induction hypothesis, the resulting solution $\hat{Y}$ for instance $\widetilde{X}$ is $\widetilde{T}$-special, and, for every strip $j \in [k]$, the resulting solution Y_j for instance X_j is T_j-special. Since both $\widetilde{T}$ and every tree $\left\{T_j\right\}_{j\in[k]}$ are subtrees of T, and since the points of Z lie on boundaries of strips in $\left\{S_j\right\}_{j\in[k]}$, the final solution Y^* is T-special. □

We next turn to prove that the solution Y^* computed by Algorithm RecursiveBST(X, T, ρ) is feasible. In order to do so, we will use the following immediate observation.

Observation 6.8. Let Y^* be the solution returned by Algorithm RecursiveBST(X, T, ρ), and let $j \in [k]$ be any strip index. Then:

- Any point $y \in Y^*$ that lies in the interior of S_j must lie on an active row of instance X_j.
- Any point $y \in Y^*$ that lies on the boundary of S_j must belong to in $\hat{Y} \cup Z$. Moreover, the points of $\hat{Y} \cup Z$ may not lie in the interior of S_j.
- If R is an active row for instance X_j, then set Z contains two points, lying on the intersection of R with the left and the right boundaries of S_j, respectively.

We are now ready to prove that the algorithm returns feasible solutions. In the following proof, when we say that a row R is an active row of strip S_j (or equivalently of instance X_j), we mean that some (input) point of instance X_j lies on row R.

Theorem 6.9. *Assume that the recursive calls to Algorithm* RECURSIVEBST *return a feasible special solution $\hat{Y}$ for instance $\widetilde{X}$, and for each $i \in [k]$, a feasible special solution Y_j for the strip instance X_j. Then the point set $Y^* = Z \cup \hat{Y} \cup (\bigcup_{j \in [k]} Y_j)$ is a feasible solution for instance X.*

Proof. It would be convenient for us to consider the set of all points in $\widetilde{X} \cup X \cup Y^*$ simultaneously. In order to do so, we start with the set $X \cup Y^*$ of points. For every $j \in [k]$, we select an arbitrary active column C_j in strip S_j, and we then add a copy of every point $p \in X_j$ to column C_j. The resulting set of points, obtained after processing all strips S_j is identical to the set $\widetilde{X}$ of points (except possibly for horizontal spacing between active columns), and we do not distinguish between them.

Consider any pair of points p, q that lie in $Y^* \cup X$, which are not aligned. Our goal is to prove that some point $r \neq p, q$ with $r \in X \cup Y^*$ lies in $\square_{p,q}$. We assume w.l.o.g. that p lies to the left of q. We also assume that $p.y < q.y$ (that is, point p is below point q); the other case is symmetric.

Assume first that at least one of the two points (say p) lies in the interior of a strip S_j, for some $j \in [k]$. We then consider two cases. First, if q also lies in the interior of the same strip, then $p, q \in X_j \cup Y_j$, and, since we have assumed that Y_j is a feasible solution for instance X_j, the two points are satisfied in $X_j \cup Y_j$, and hence in $X \cup Y^*$. Otherwise, q does not lie in the interior of strip S_j. Then, from Observation 6.8, if R is the row on which point p lies, then R is an active row for instance X_j, and the point that lies on the intersection of the row R and the right boundary of strip S_j was added to Z. This point satisfies the pair (p, q).

Therefore, we can now assume that both p and q lie on boundaries of strips $\{S_j \mid j \in [k]\}$. Since every pair of consecutive strips share a boundary, and since, from the above assumptions, p lies to the left of q, we can choose the strips S_j and S_l, such that p lies on the left boundary of S_j, and q lies on the right boundary of S_l. Notice that it is possible that $S_j = S_l$.

Notice that, if p lies on a row that is active for strip S_j, then a point that lies on the same row and belongs to the right boundary of the strip S_j has been added to Z; this point satisfies the pair (p, q). Similarly, if q lies on a row that is active for strip S_l, the pair (p, q) is satisfied by a point of Z that lies on the same row and belongs to the left boundary of S_l.

Therefore, it remains to consider the case where point p lies on a row that is inactive for S_j, and point q lies on a row that is inactive for S_l. From Observation 6.8, both p and q belong to $\hat{Y} \cup Z$.

The following observation will be useful for us. Recall that the points of $\widetilde{X}$ are not included in $X \cup Y^*$.

Observation 6.10. Assume that there is some point $r \neq p, q$, such that $r \in \square_{p,q}$, and $r \in \widetilde{X}$. Then the pair (p, q) is satisfied in set $X \cup Y^*$.

Proof. Since $r \in \widetilde{X}$, and $r \in \square_{p,q}$, there must be some strip S_i, that lies between strips S_j and S_l (where $j \leq i \leq l$), such that point r lies on the column C_i (recall that this is the unique active column of S_i that may contain points of $\widetilde{X}$). But then the row R to which point r belongs is an active row for strip S_i. Therefore, two points, lying on the intersection of R with the two boundaries of strip S_i were added to Z, and at least one of these points must lie in $\square_{p,q}$. Since $Z \subseteq Y^*$, the observation follows. $\square$

From the above observation, it is sufficient to show that some point $r \in \hat{Y} \cup Z \cup \widetilde{X}$ that is distinct from p and q, lies in $\square_{p,q}$. We distinguish between three cases.

The first case happens when $p, q \in \hat{Y}$. Since set $\hat{Y}$ is a feasible solution for instance $\widetilde{X}$, there is some point $r \in \square_{p,q}$ that is distinct from p and q, and lies in $\hat{Y} \cup \widetilde{X}$.

The second case happens when neither p nor q lie in $\hat{Y}$, so both $p, q \in Z$. Consider strip S_{j-1} lying immediately to the left of strip S_j. Since p lies on a row that is inactive for strip S_j, but $p \in Z$, such a strip must exist, and moreover, the row R to which p belongs must be active for strip S_{j-1}. Therefore, the point lying on the intersection of the column C_{j-1} (the unique active column of S_{j-1} containing points of $\widetilde{X}$) and R belongs to $\widetilde{X}$; we denote this point by p'.

Similarly, consider strip S_{l+1} lying immediately to the right of S_l. Since q lies on a row that is inactive for strip S_l, but $q \in Z$, such a strip must exist, and moreover, the row R' to which q belongs must be active for strip S_{l+1}. Therefore, the point lying on the intersection of C_{l+1} and R' belongs to $\widetilde{X}$; we denote this point by q'.

Since the set $\widetilde{X} \cup \hat{Y}$ of points is satisfied, some point $r \in \widetilde{X} \cup \hat{Y}$ that is distinct from p' and q', lies in $\square_{p',q'}$. Moreover, from Observation 2.3, we can choose this point so that it lies on the boundary of the rectangle $\square_{p',q'}$. Assume first that r lies on the left boundary of this rectangle. Then, since $\hat{Y}$ is a special solution for instance $\widetilde{X}$, $r \in \widetilde{X}$ must hold. If R'' denotes the row on which r lies, then R'' is an active row for strip S_{j-1}, and so a point that lies on the intersection of row R'' and the right boundary of S_{j-1} belongs to Z. That point satisfies $\square_{p,q}$. The case where r lies on the right boundary of $\square_{p',q'}$ is treated similarly.

Assume now that r lies on the top or the bottom boundary of $\square_{p',q'}$, but not on one of its corners. Then, since solution $\hat{Y}$ is special for $\widetilde{X}$, point r must lie in the rectangle $\square_{p,q}$. Moreover, since we have assumed that neither p nor q lie in $\hat{Y}$, $r \neq p, q$. But then $r \in \widetilde{X} \cup \hat{Y}$ lies in $\square_{p,q} \setminus \{p, q\}$ and by Observation 6.10, pair (p, q) is satisfied in $X \cup Y^*$.

The third case happens when exactly one of the two points (say p) lies in $\hat{Y}$, and the other point does not lie in $\hat{Y}$, so $q \in Z$ must hold. We define the point $q' \in \widetilde{X}$ exactly as in the second case. Since $p, q' \in \widetilde{X} \cup \hat{Y}$, there must be a point $r \in \square_{p,q'} \setminus \{p, q'\}$ that lies in $\widetilde{X} \cup \hat{Y}$. From Observation 6.8, we can choose the point r, so that it lies on the left or on the bottom boundary of $\square_{p,q'}$ (that is, its x- or its y-coordinate is aligned with the point p). If r lies on the left boundary of $\square_{p,q'}$, then it also lies in $\square_{p,q}$, and from Observation 6.10, pair (p, q) is satisfied in $X \cup Y^*$. If it lies on the bottom boundary of $\square_{p,q'}$, but it is not the bottom right corner of the rectangle, then, using the same reasoning as in Case 2, it must lie in $\square_{p,q}$, and it is easy to see that $r \neq q$. Lastly, if r is the bottom right corner of $\square_{p,q'}$, then $r \in \widetilde{X}$. As before, there is a copy of r, that lies on the left boundary of strip S_{l+1} and belongs to Z, that satisfies the pair (p, q). $\square$

6.6 Leaf instances (proof of Theorem 6.4)

The goal of this subsection is to prove Theorem 6.4. For convenience, given an input instance X of Min-Sat, we denote $r(X) = m$ and $c(X) = n$. Our goal is to compute an optimal canonical solution Y for X in time $\text{poly}(m) \cdot n^{O(n)}$. The solution can then be turned into a special one using the following observation.

Observation 6.11. There is an algorithm, that, given a set X of points that is a semi-permutation, and a canonical solution Y for X, computes a special solution Y' for X, such that $|Y'| \le 2|X|+2|Y|$.

Proof. We construct Y' as follows: For each point $p \in X \cup Y$, we add two points, p' and p'' to Y', whose y-coordinate is the same as that of p, such that p' and p'' lie on the lines of $\mathcal{L}$ appearing immediately to the left and immediately to the right of p, respectively. It is easy to verify that $|Y'| = 2|X| + 2|Y|$ and that Y' is a special solution.

We now verify that $X \cup Y'$ is a satisfied set of points. Consider two points p, q in $X \cup Y'$, such that $p.x < q.x$. Notice that p is either an original point in X or it is a copy of some point $\hat{p} \in X \cup Y$. If $p \in X$ or $p = \hat{p}'$ for $\hat{p} \in X \cup Y$, then the point $\hat{p}''$ lies in the rectangle $\square_{p,q}$. Therefore, we can assume that $p = \hat{p}''$ for some point $\hat{p} \in X \cup Y$. By a similar reasoning, we can assume that $q = \hat{q}'$ for some point $\hat{q} \in X \cup Y$. Since $X \cup Y$ is a satisfied point set, there must be a point $r \in X \cup Y$ that lies in the rectangle $\square_{\hat{p},\hat{q}}$. From Observation 2.3, we can choose r such that either $r.x = \hat{p}.x$ or $r.y = \hat{p}.y$. In either case, point r'' also lies in $\square_{\hat{p}'',\hat{q}'} = \square_{p,q}$, so (p, q) is satisfied by $X \cup Y'$. Therefore, $X \cup Y'$ is a satisfied set of points. □

This observation allows us to turn the optimal canonical solution into a special solution of cost at most $2|X| + 2|Y| \le 2|X| + 2\mathsf{OPT}(X)$, in time $\text{poly}(|X|)$. We start by providing several definitions and structural observations that will be helpful in designing the algorithm.

6.6.1 Conflicting sets

Our algorithm uses the notion of conflicting point sets, defined as follows.

Definition 6.12 (Conflicting Sets)**.** Let Z and Z' be sets of points. We say that Z and Z' are *conflicting* if $Z \cup Z'$ is not satisfied.

The following definition is central to our algorithm.

Definition 6.13 (Top representation)**.** Let Z be any set of points. A *top representation* of Z, that we denote by $\text{top}(Z)$, is a subset $Z' \subseteq Z$ of points, obtained as follows: for every column C that contains points from Z, we add the topmost point of Z that lies on C to Z'.

Observation 6.14. Let $\text{top}(Z) \subseteq Z$ be the top representation of Z, and let R be a row lying strictly above all points in Z. Let Y be any set of points lying on row R. Then $top(Z)$ is conflicting with Y if and only if Z is conflicting with Y.

Proof. Assume that Y is conflicting with Z, and let $p \in Y$, $q \in Z$ be a pair of points, such that no point of $Y \cup Z$ lies in $\square_{p,q} \setminus \{p, q\}$. But then $q \in \mathrm{top}(Z)$ must hold, and no point of $\mathrm{top}(Z) \cup Y$ lies in $\square_{p,q} \setminus \{p, q\}$, so $\mathrm{top}(Z)$ and Y are conflicting. Assume now that $\mathrm{top}(Z)$ and Y are conflicting, and let $p \in Y$, $q \in \mathrm{top}(Z)$ be a pair of points, such that no point of $Y \cup \mathrm{top}(Z)$ lies in $\square_{p,q} \setminus \{p, q\}$. But then no point of $Y \cup Z$ lies in $\square_{p,q} \setminus \{p, q\}$, and, since $\mathrm{top}(Z) \subseteq Z$, sets $\mathrm{top}(Z)$ and Y are conflicting. $\square$

6.6.2 The setup

Let X be the input point set, that is a semi-permutation. We denote by $\mathcal{R} = \{R_1, \ldots, R_m\}$ the set of all active rows for X, and we assume that they are indexed in their natural bottom-to-top order. We denote by $\mathcal{C} = \{C_1, \ldots, C_n\}$ the set of all active columns of X, and we assume that they are indexed in their natural left-to-right order. We also denote $X = \{p_1, \ldots, p_m\}$, where for all $1 \leq i \leq m$, p_i is the unique point of X lying on row R_i. For an index $1 \leq t \leq m$, we denote by $\mathcal{R}_{\leq t} = \{R_1, \ldots, R_t\}$, and we denote by $X_{\leq t} = \{p_1, \ldots, p_t\}$.

Note that, if Y is a feasible solution to instance X, then for all $1 \leq t \leq m$, the set $X_{\leq t} \cup Y_{\leq t}$ of points must be satisfied (here $Y_{\leq t}$ is the set of all points of Y lying on rows of $\mathcal{R}_{\leq t}$.) Our dynamic programming-based algorithm constructs the optimal solution row-by-row, using this observation. We use height profiles, that we define next, as the "states" of the dynamic program.

A *height profile* π assigns, to every column $C_i \in \mathcal{C}$, a value $\pi(C_i) \in \{1, \ldots, n, \infty\}$. Let Π be the set of all possible height profiles, so $|\Pi| \leq n^{O(n)}$. For a profile $\pi \in \Pi$, we denote by $M(\pi)$ the largest value of $\pi(C_i)$ for any column $C_i \in \mathcal{C}$ that is not ∞. Given a height profile π, let $\mathcal{C}(\pi) \subseteq \mathcal{C}$ be the set of columns C_i with $\pi(C_i) < \infty$, and let $\mathcal{C}'(\pi) = \mathcal{C} \setminus \mathcal{C}(\pi)$. We can then naturally associate an ordering ρ_π of the columns in $\mathcal{C}(\pi)$ with π as follows: for columns $C_i, C_j \in \mathcal{C}(\pi)$, C_i appears before C_j in π iff either (i) $\pi(C_i) < \pi(C_j)$; or (ii) $\pi(C_i) = \pi(C_j)$ and $i < j$.

Consider now any point set Z, where every point lies on a column of $\mathcal{C}$. Let $Z' = \mathrm{top}(Z)$, and let $\sigma(Z)$ denote the ordering of the points in Z', such that p appears before p' in σ iff either (i) $p.y < p'.y$; or (ii) $p.y = p'.y$ and $p.x < p'.x$. Consider now any profile $\pi \in \Pi$. We say that point set Z is *consistent with profile* π (see Figure 8 for an illustration) iff the following hold:

- For every column $C_i \in \mathcal{C}$, if $\pi(C_i) = \infty$, then no point of Z lies on C_i; and
- For all $1 \leq i \leq |Z'|$, the ith point in $\sigma(Z)$ lies on the ith column of ρ_π.

6.6.3 Our DP

Consider some integer $t : 1 \leq t \leq m$ (viewed as a row index) and some height profile π; we define $M(\pi)$ as the largest value of $\pi(C_j)$ for $C_j \in \mathcal{C}$ that is not ∞. We say that π is a *legal profile* for time t iff $M(\pi) \leq t$, and, if C_i is the column containing the input point p_t, then $\pi(C_i) = M(\pi)$

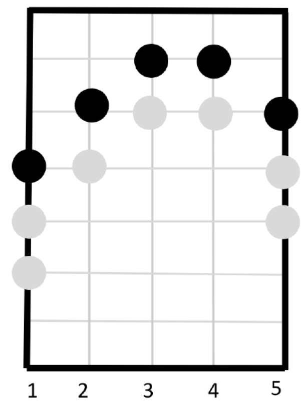


Figure 8: An illustration of height profile that is consistent with a set Z of points. The set $\mathrm{top}(Z)$ is shown by dark points. The height profile is $\pi(C_1) = 1, \pi(C_2) = \pi(C_5) = 2$ and $\pi(C_3) = h_Z(C_4) = 3$.

(that is, column C_i has the largest value $\pi(C_i)$ that is not ∞; we note that it is possible that other columns $C_j \neq C_i$ also have $\pi(C_j) = M(\pi)$).

For every integer $1 \leq t \leq m$, and every height profile π that is legal for t, there is an entry $T[t, \pi]$ in the dynamic programming table. The entry is supposed to store the minimum-cardinality set Z of points with the following properties:

- $X_{\leq t} \subseteq Z$;
- all points of Z lie on rows $R_1, \ldots, R_t$;
- Z is a satisfied point set; and
- Z is consistent with π.

Clearly, there number of entries in the dynamic programming table is bounded by $mn^{O(n)}$. We fill out the entries in the increasing order of the index t.

We start with $t = 1$. Consider any profile π that is legal for time t. Recall that for every column $C_j \in \mathcal{C}$, $\pi(C_j) \in \{1, \infty\}$ must hold, and moreover, if C_i is the unique column containing the point $p_1 \in X$, then $\pi(C_i) = 1$ must hold. We let $T[1, \pi]$ contain the following set of points: for every column $C_j \in \mathcal{C}$ with $\pi(C_j) = 1$, we add the point lying in the intersection of column C_j and row R_1 to the point set stored in $T[t, \pi]$. It is immediate to verify that the resulting point set is consistent with π, it is satisfied, it contains $X_{\leq 1} = \{p_1\}$, and it is the smallest-cardinality point set with these properties.

We now assume that for some $t \geq 1$, we have computed correctly all entries $T[t', \pi']$ for all $1 \leq t' \leq t$, and for all profiles π' legal for t'. We now fix some profile π that is legal for $t + 1$, and show how to compute entry $T[t+1,\pi]$.

Let $r = M(\pi)$ (recall that this is the largest value of $\pi(C_j)$ that is not ∞), and let $\mathcal{C}_1 \subseteq \mathcal{C}$ be the set of all columns C_j with $\pi(C_j) = r$. Recall that, if C_j is the column containing the input point p_{t+1}, then $C_j \in \mathcal{C}_1$ must hold. Let P be the set of $|\mathcal{C}_1|$ points that lie on the intersection of the row R_{t+1} and the columns in $\mathcal{C}_1$.

Consider now any profile π' that is legal for t, and let $\hat{Z} = T[t,\pi']$. Denote $\hat{Z}' = \mathrm{top}(\hat{Z})$. We say that profile π' is a *candidate profile* if (i) π' is legal for t; (ii) the point sets $P, \hat{Z}'$ do not conflict; (iii) $\mathcal{C}'(\pi') \subseteq \mathcal{C}'(\pi) \cup \mathcal{C}_1$; and (iv) if we discard from ρ_π and from $\rho_{\pi'}$ the columns of $\mathcal{C}_1$, then the two orderings are defined over the same set of columns and they are identical. We select a candidate profile π' that minimizes $|T[t,\pi']|$, and let $Z = \hat{Z} \cup P$, where $\hat{Z}$ is the point set stored in $T[t,\pi']$. We then set $T[t+1,\pi] = Z$.

We now verify that set Z has all required properties. If the entry $T[t,\pi']$ was computed correctly, then point set $\hat{Z}$ is satisfied. Since $\mathrm{top}(\hat{Z})$ and P are not conflicting, from Observation 6.14 neither are $\hat{Z}$ and P. Therefore, set Z is satisfied. If the entry $T[t,\pi']$ was computed correctly, then $X_{\le t} \subseteq \hat{Z}$. Since $p_{t+1} \in P$, we get that $X_{\le(t+1)} \subseteq Z$.

Next, we show that Z is consistent with the profile π. Let $Z' = \mathrm{top}(Z)$, and consider the ordering $\sigma(Z)$ of the points in Z'. It is easy to verify that the last $|P|$ points in this ordering are precisely the points of P. The remaining points in this ordering can be obtained from the point set $\hat{Z}'$, by first ordering them according to ordering $\sigma(\hat{Z})$, and then deleting all points lying on columns of $\mathcal{C}_1$. From the definition of candidate profiles, it is easy to verify that for all $1 \le i \le |Z'|$, the ith point in $\sigma(Z)$ lies on the ith column of ρ_π. Therefore, Z is consistent with profile π.

Lastly, it remains to show that the cardinality of Z is minimized among all sets with the above properties. Let Z^* be the point set that contains $X_{\le t+1}$, is satisfied, is consistent with profile π, with every point of Z^* lying on rows $R_1,\ldots,R_{t+1}$, such that $|Z^*|$ is minimized among all such sets. It is easy to verify that the set of points of Z^* lying on row R_{t+1} must be precisely P. This is since point p_{t+1} must belong to Z^*, and so every column C_i with $\pi(C_i) = M(\pi)$ must have a point lying on the intersection of C_i and R_{t+1} in Z^*. Let $\hat{Z} = Z^* \setminus P$. Clearly, $\hat{Z}$ is satisfied, all points of $\hat{Z}$ lie on rows $R_1,\ldots,R_t$, and $X_{\le t} \subseteq \hat{Z}$. Moreover, there is no conflict between $\mathrm{top}(\hat{Z})$ and P.

We define a new height profile π' as follows: for every column $C_i \in \mathcal{C}$, if no point of $\hat{Z}$ lies on C_i, then $\pi'(C_i) = \infty$. Otherwise, let p be the unique point in $\hat{Z}'$ that lies on C_i, and assume that it lies on row $R_{t'}$. Then we set $\pi'(C_i) = t'$. Notice that $\mathcal{C}'(\pi') \subseteq \mathcal{C}'(\pi) \cup \mathcal{C}_1$, and, if we discard from ρ_π and from $\rho_{\pi'}$ the columns of $\mathcal{C}_1$, then the two orderings are defined over the same set of columns and they are identical.

It is easy to verify that the resulting profile π' must be legal for t. Therefore, profile π' was considered by the algorithm. Since $\hat{Z}'$ and P are not conflicting, it is easy to verify that for any other set $\tilde{Z}$ of points that lie on rows $R_1,\ldots,R_t$ and are consistent with profile π', set $\mathrm{top}(\tilde{Z})$ does not conflict with P. Therefore, π' must be a candidate profile for π. We conclude that

$|T[t, \pi']| \leq |\hat{Z}|$, and $|T[t, \pi]| \leq |T[t, \pi']| + |P| = |Z^*|$, so $|Z| \leq |Z^*|$ must hold.

The output of the algorithm is a set $T[m, \pi] \setminus X$ of smallest cardinality among all profiles $\pi \in \Pi$ that are consistent with m. It is immediate to verify that this is a feasible and optimal solution for X.

As observed before, the number of entries in the dynamic programming table is $m \cdot n^{O(n)}$, and computing each entry takes time $n^{O(n)}$. Therefore, the total running time of the algorithm is bounded by $m \cdot n^{O(n)}$.

7 An $O(\log \log n)$-competitive online algorithm

In this section we extend the $O(\log \log n)$-approximation algorithm from the previous section to the online setting, completing the proof of Theorem 1.2. To this end, the recursive algorithm is not quite convenient to work with. In Subsection 7.1, we present an equivalent iterative description of our algorithm. In Subsection 7.2, we slightly modify the solution Y that the algorithm returns to obtain another solution $\hat{Y}$ that is more friendly for the online setting, before presenting the final online algorithm in Subsection 7.3.

7.1 Unfolding the recursion

Let X be an input set of points that is semi-permutation, with $|X| = m$, and $c(r) = n$. Let T be a balanced partitioning tree of height $H = O(\log n)$ for X.

We now construct another tree R, that is called a *recursion tree*, and which is unrelated to the tree T. Every vertex q of the tree R is associated with an instance $\mathcal{I}(q)$ of Min-Sat that arose during the recursive execution of Algorithm RecursiveBST(X, T, ρ), with $\rho = 1$. Specifically, for the root r of the tree R, we let $\mathcal{I}(r) = X$. For every vertex q of the tree R, if Algorithm RecursiveBST(X, T, ρ), when called for instance $\mathcal{I}(q)$, constructed instances $\mathcal{I}_1, \ldots, \mathcal{I}_z$ (recall that one of these instances is a compressed instance, and the remaining instances are strip instances), then vertex q has z children in tree R, each of which is associated with one of these instances.

For a vertex $q \in V(R)$, let $n(q)$ be the number active columns in the instance $\mathcal{I}(q)$. Recall that instance $\mathcal{I}(q)$ corresponds to some subtree of the partitioning tree T, that we denote by T_q. For all $i \geq 0$, let Λ'_i be the set of all vertices of the tree R that lie at distance exactly i from the root of R. We say that the vertices of Λ'_i belong to the ith layer of R. Notice that, if vertex q lies in the ith layer of R, then the height of the corresponding tree T_q is bounded by $H \cdot (2/3)^i$ (the constant $2/3$ is somewhat arbitrary and is used because, when a tree T' is split in its middle layer, each of the resulting subtrees has height at most $\lfloor \text{height}(T')/2 \rfloor + 1 \leq 2H/3$). Recall that the recursion terminates once we obtain instances whose corresponding partitioning trees have height 1. It is then easy to verify that the height of the recursion tree R is bounded by $O(\log \log n)$.

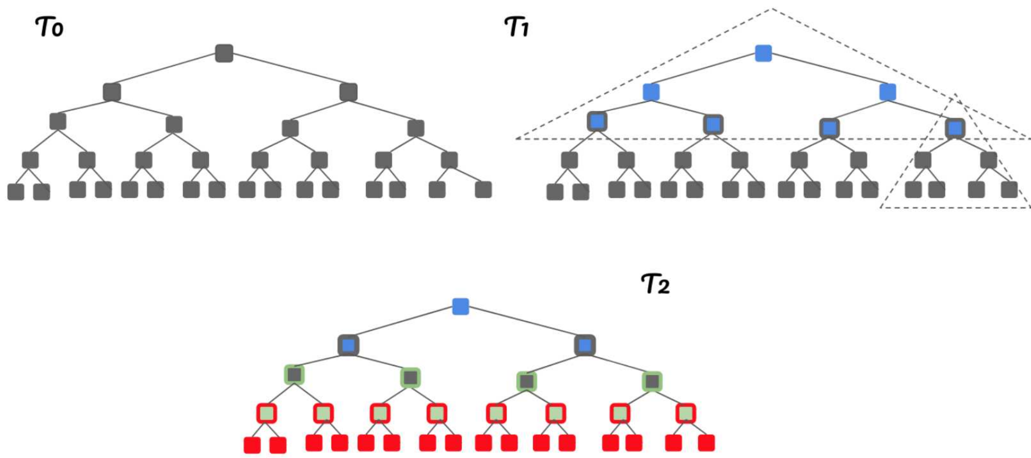


Figure 9: An illustration of the families $\mathcal{T}_i$. Notice that $\mathcal{T}_1$ contains 5 trees, each having height 2.

Consider now some layer Λ_i of the recursion tree R. We let $\mathcal{T}_i$ be the collection of all subtrees of the partitioning tree T corresponding to the vertices of Λ_i, so $\mathcal{T}_i = \left\{T_q \mid q \in \Lambda_i\right\}$. Recall that all trees in $\mathcal{T}_i$ have height at most $H \cdot (2/3)^i$.

Notice that $\mathcal{T}_0 = \{T\}$. Let $U = \{v_1, \ldots, v_k\}$ be the middle layer of T. Then $\mathcal{T}_1 = \left\{T_{v_1}, \ldots, T_{v_k}, T_r\right\}$. Set $\mathcal{T}_2$ is similarly obtained by subdividing every tree in $\mathcal{T}_1$, and so on. (See Figure 9 for an illustration.) The construction of the tree sets $\mathcal{T}_i$ can be described using the following process:

- Start from $\mathcal{T}_0 = \{T\}$.
- For $i = 1, \ldots, D$, if some tree in $\mathcal{T}_{i-1}$ has height greater than 1, then construct the set $\mathcal{T}_i$ of trees as follows: For every tree $T' \in \mathcal{T}_i$ whose height is greater than 1, consider the split partitioning trees $\left\{\{T_j\}, \widetilde{T}\right\}$ by the (boundaries of) middle-layer strips. Notice that $\widetilde{T}$ is rooted at the root of T' and each T_j at a leaf of $\widetilde{T}$. These subtrees are added into $\mathcal{T}_i$.

The following observation is immediate.

Observation 7.1. For each $1 \leq i \leq D$, $\bigcup_{T' \in \mathcal{T}_i} V(T') = V(T)$, and for each pair $T', T'' \in \mathcal{T}_i$ of trees, either $V(T') \cap V(T'') = \emptyset$ or the root of one of these trees is a leaf of the other tree.

7.1.1 Boxes

Fix an index $1 \leq i \leq D$, and consider any tree $T' \in \mathcal{T}_i$. Denote the number of leaves by k, and let $v_1, \ldots, v_k$ be the leaves of T'. If v is the root vertex of the tree T', then we can view T' as defining a hierarchical partitioning scheme of the strip $S(v)$, until we obtain the partition

$(S(v_1), S(v_2), \dots, S(v_k))$ of $S(v)$. We define a collection of T'-boxes as follows. Let $X' = X \cap S(v)$ be the set of all points lying in strip $S(v)$. Assume that $X' = \{p_1, p_2, \dots, p_{m'}\}$, where the points are indexed in the increasing order of their y-coordinates.

We now iteratively partition the point set X' into boxes, where each box is a consecutive set of points of X'. Let v_i be the leaf vertex of T', such that $p_1 \in S(v_i)$, and let m_1 be the largest index, such that all points $p_1, \dots, p_{m_1}$ lie in $S(v_i)$. We then then define a box $B_1 = \{p_1, p_2, \dots, p_{m_1}\}$. We discard the points $p_1, \dots, p_{m_1}$, and continue this process to define B_2 (starting from point $p_{m_1} + 1$), and so on, until every point of X' belongs to some box.

We let $\mathcal{B}(T') = \left\{B_1, B_2, \dots, B_{z(T')}\right\}$ be the resulting partition of the points in X', where we refer to each set B_i as a T'-box, or just a box. For each box $B' \in \mathcal{B}(T')$, the lowest and the highest rows containing points of B' are denoted by $\mathsf{first}(B')$ and $\mathsf{last}(B')$ respectively.

7.1.2 Projections of points

Recall that the solution that our recursive algorithm returns is T-special, that is, the points that participate in the solution lie on the active rows of X and on T-auxiliary columns. Let Y be a feasible solution obtained by our algorithm RecursiveBST(X, T, ρ), where $\rho = 1$. Notice the every point in Y is obtained by "projecting" some input point $p \in X$ to the boundary of some strip $S(v)$ for $v \in V(T)$, where strip $S(v)$ contains p. For each point $p \in X$ and node $v \in V(T)$ of the partitioning tree, such that $p \in S(v)$, we define the set $\mathsf{proj}(p, v)$ to contain two points on the same row as p, that lie on the left and right boundaries of $S(v)$, respectively. We denote by $Y_{p,v}$ the set that contain the two points $\mathsf{proj}(p, v)$ if our algorithm adds these two points to the solution. Since all points of X lie on distinct rows, for any two points $p \neq p'$, for any pair $v, v' \in V(T)$ of vertices, $\mathsf{proj}(p, v) \cap \mathsf{proj}(p', v') = \emptyset$. We can now write the solution Y as $Y = \bigcup_{p\in X} \bigcup_{v\in V(T)} Y_{p,v}$. The following lemma characterizes the points of Y in terms of boxes.

Lemma 7.2. *For every point $p \in X$ and every node $v \in V(T)$ of the partitioning tree, $|Y_{p,v}| = 2$ if and only if there is an index $1 \le i \le D$, and a subtree $T' \in \mathcal{T}_i$, such that (i) p is the first or the last point of its T'-box; (ii) v lies in the middle layer of T'; and (iii) $S(v)$ contains p.*

Proof. Let $1 \le i \le D$ be an index, and let $T' \in \mathcal{T}_i$ be the corresponding partitioning tree. We denote the corresponding point set by X'. Let $v_1, \dots, v_k$ be the leaf vertices of T', and let $u_1, \dots, u_r$ be the vertices lying in the middle layer of T'. The only points added to the solution when instance X' is processed are the following: for every $1 \le j \le r$, for every point $p \in X' \cap S(u_j)$, we add the two copies of p to the boundaries of $S(u_j)$. In subsequent, or in previous iterations, we may add points to boundaries of $S(u_j)$, but we will never add two copies of the same input point to both boundaries of $S(u_j)$. Therefore, if $|Y_{p,v}| = 2$, then there must be an index i, and a tree $T' \in \mathcal{T}_i$, such that v lies in the middle layer of T', and $S(v)$ contains p. Observe that for every T'-box B, instance X' only contains the first and the last point of B; all remaining points are redundant for X', and such points are not projected to the boundaries of their strips. Therefore, p must be the first or the last point of its T'-box. □

7.1.3 An equivalent view of our algorithm

We can think of our algorithm as follows. First, we compute all families $\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_D$ of trees, and for each tree T' in each family $\mathcal{T}_i$, all T'-boxes. For each point $p \in X$, for each vertex $v \in V(T)$ of the partitioning tree T, such that $p \in S(v)$, we add the projection points $\mathsf{proj}(p, v)$ to the solution, depending on whether there exists an index $i \in \{0, 1, \ldots, D\}$ and a tree $T' \in \mathcal{T}_i$, such that v lies in the middle layer $U_{T'}$ of T', and whether p is the first or last point of its T'-box. Notice that in the online setting, when the point p arrives, we need to be able to immediately decide which copies of p to add to the solution. Since the trees in the families $\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_D$ are known in advance, and we discover, for every vertex $v \in V(T)$ whether $p \in S(v)$ immediately when p arrives, the only missing information, for every relevant tree T', is whether vertex p is the first or the last vertex in its T'-box. In fact it is easy to check whether it is the first vertex in its T'-box, but we will not discover whether it is the last vertex in its T'-box until the next iteration, at which point it is too late to add points to the solution that lie in the row of p.

In order to avoid this difficulty, we slightly modify the instance and the solution.

7.2 Online-friendly solutions

In this section we slightly modify both the input set of points and the solutions produced by our algorithm, in order to adapt them to the online setting.

7.2.1 Modifying the instance

Let $X = \{p_1, \ldots, p_m\}$ be the input set of points, where for all $1 \le i \le m$, the y-coordinate of p_i is i. We produce a new instance $X' = \left\{p'_i, p''_i \mid 1 \le i \le m\right\}$, as follows: for all $1 \le i \le m$, we let p'_i and p''_i be points whose y-coordinates are $(2i-1)$ and $2i$, respectively, and whose x-coordinate is the same as that of p_i. We refer to p'_i and to p''_i as *copies of* p_i.

Clearly, $|X'| = 2|X|$, and it is easy to verify that $\mathsf{OPT}(X') \le 2\mathsf{OPT}(X)$. Indeed, if we let Y be a solution for X, we can construct a solution Y' for X', by creating, for every point $q \in Y$, two copies q' and q'', that are added to rows with y-coordinates $2q.y - 1$ and $2q.y$ respectively.

For convenience, we denote $\mathcal{T} = \bigcup_{i=0}^{D} \mathcal{T}_i$. Notice that, for every tree $T' \in \mathcal{T}$, for every point p_i of the original input X, copy p'_i of p_i may never serve as the last point of a T'-box, and copy p''_i of p_i may never serve as the first point of a T'-box.

7.2.2 Modifying the solution

Let Y be the solution that our $O(\log\log n)$-approximation algorithm produces for the new instance X'. For convenience, for all $1 \le i \le 2m$, we denote by R_i the row with y-coordinate i. Notice that all points of Y lying on a row R_{2i-1} are projections of the point p'_i. Since this point may only serve as the first point of a T'-box for every tree $T' \in \mathcal{T}$, when point p_i' arrives online, we can immediately compute all projections of p_i' that need to be added to the solution. All points of Y lying on row R_{2i} are projections of the point p_i''. This point may only serve as the last point of a T'-box in a tree $T' \in \mathcal{T}$. But we cannot know whether p_i'' is the last point in its T'-box until we see the next input point. Motivated by these observations, we now modify the solution Y as follows.

We perform m iterations. In iteration i, we consider the row R_{2i}. If no point of Y lies on row R_{2i}, then we continue to the next iteration. Otherwise, we move every point of Y that lies on row R_{2i} to row R_{2i+1} (that is, one row up). Additionally, we add another copy p_i''' of point p_i to row R_{2i}, while preserving its x-coordinate.

In order to show that the resulting solution is a feasible solution to instance X', it is sufficient to show that the solution remains feasible after every iteration. Let Y_i be the solution Y obtained before the ith iteration, and let $S_i = X' \cup Y_i$. We can obtain the new solution Y_{i+1} equivalently as follows. First, we collapse the rows R_{2i+1} and R_{2i} for the set S_i of points into the row R_{2i+1}, obtaining a new set S_i' of points that is guaranteed to be satisfied. Notice that now both row R_{2i+1} and row R_{2i-1} contain a point at x-coordinate $(p_i).x$, while row R_{2i} contains no points. Therefore, if we add to S_i' a point with x-coordinate $(p_i).x$, that lies at row R_{2i}, then the resulting set of points, that we denote by S_{i+1} remains satisfied. But it is easy to verify that $S_{i+1} = X' \cup Y_{i+1}$, where Y_{i+1} is the solution obtained after iteration i. We denote by Y' the final solution obtained by this transformation of Y. It is easy to see that $|Y'| \leq 2|Y| \leq O(\log\log n)\mathsf{OPT}(X)$.

7.3 The final online algorithm

Let $X = \{p_1, \ldots, p_m\}$ be an input set of points. We will describe the online algorithm that produces a feasible solution to instance X. This algorithm would mimic the behavior of the point set Y' as described earlier.

Initially, the algorithm computes the collection $\mathcal{T}$ of trees before any input arrives. Now, in iteration i, when input p_i arrives, we compute $FIRST(i)$:

$$FIRST(i) = \bigcup_{T' \in \mathcal{T} : p_i \text{ is first of a } T'\text{-box}} \left(\bigcup_{v : v \text{ is middle layer of } T'\text{, and } S(v) \text{ contains } p_i} \mathsf{proj}(p_i, v) \right)$$

We also compute $LAST(i-1)$:

$$LAST(i-1) = \bigcup_{T' \in \mathcal{T} : p_{i-1} \text{ is last of a } T'\text{-box}} \left(\bigcup_{v : v \text{ is middle layer of } T' \text{and } S(v) \text{ contains } p_{i-1}} \mathsf{proj}(p_{i-1}, v) \right)$$

We add copies of $FIRST(i)$ and $LAST(i-1)$, and a copy of p_{i-1} points on row i. It is easy to verify that the resulting solution is precisely the solution obtained after collapsing every two consecutive rows in Y' (as discussed previously). Therefore the solution is feasible and has cost at most $|Y'| \leq O(\log\log n)\mathsf{OPT}$.

8 Wilber and Guillotine bounds

In this section, we provide an alternative, geometric proof of the fact that $\mathsf{WB}(X) \leq 2\mathsf{OPT}(X)$, which can be extended to the Guillotine bound. The original proof of Wilber [29] is done in the tree view.

8.1 Wilber bound

It is sufficient to prove that, if X is a semi-permutation, and T is any partitioning tree for X, then $\mathsf{WB}_T(X) \leq 2\mathsf{OPT}(X)$.

We prove this claim by induction on the height of T. The base case, when the height of T is 0, is obvious: there is only one active column, and so $\mathsf{WB}_T(X) = \mathsf{OPT}(X) = 0$.

We now consider the inductive step. Let X be any point set that is a semi-permutation, and let T be any partitioning tree for X, such that the height of T is at least 1. Let v be the root vertex of T, $L = L(v)$ the line that v owns, and let v_L, v_R be the two children of v. We assume w.l.o.g. that the strip $S(v_L)$ lies to the left of $S(v_R)$. We denote $S(v) = B$ – the bounding box of the instance, $S(v_L) = S^L, S(v_R) = S^R$, and we also denote $X^L = X \cap S^L$, $X^R = X \cap S^R$. Lastly, we let T^L and T^R be the subtrees of T rooted at v_L and v_R, respectively. We prove the following claim.

Claim 8.1.

$$\mathsf{OPT}(X) \geq \mathsf{OPT}(X^L) + \mathsf{OPT}(X^R) + \mathsf{cost}(v)/2.$$

Notice that, if the claim is correct, then we can use the induction hypothesis on X^L and X^R with the trees T^L and T^R respectively, to conclude that:

$$\mathsf{OPT}(X) \geq \frac{1}{2}\left(\mathsf{WB}_{T^L}(X_L) + \mathsf{WB}_{T^R}(X_R) + \mathsf{cost}(v)\right) = \frac{1}{2}\mathsf{WB}_T(X).$$

Therefore, in order to complete the proof of Claim 2.7, it is enough to prove Claim 8.1.

Proof. Claim 8.1 Let Y be an optimal solution to instance X. We can assume w.l.o.g. that Y is a canonical solution, so no point of Y lies on the line L. Let Y^L, Y^R be the subsets of points of Y that lie to the left and to the right of the line L, respectively.

Let $\mathcal{R}^L$ be the set of all rows R, such that (i) no point of X^L lies on R, and (ii) some point of Y^L lies on R. We define a set $\mathcal{R}^R$ of rows similarly for instance X^R. The crux of the proof is the following observation.

Observation 8.2.

$$|\mathcal{R}^L| + |\mathcal{R}^R| \geq \mathsf{cost}(v)/2.$$

Before we prove Observation 8.2, we show that Claim 8.1 follows from it. In order to do so, we will define a new feasible solution $\hat{Y}^L$ for instance X^L, containing at most $|Y^L| - |\mathcal{R}^L|$ points, and similarly, we will define a new feasible solution $\hat{Y}^R$ for instance X^R, containing at most $|Y^R| - |\mathcal{R}^R|$ points. This will prove that $\mathsf{OPT}(X^L) \le |Y^L| - |\mathcal{R}^L|$ and $\mathsf{OPT}(X^R) \le |Y^R| - |\mathcal{R}^L|$, so altogether, $\mathsf{OPT}(X^L) + \mathsf{OPT}(X^R) \le |Y| - |\mathcal{R}^L| - |\mathcal{R}^R| \le \mathsf{OPT}(X) - \mathsf{cost}(v)/2$, thus proving Claim 8.1.

We now show how to construct the solution $\hat{Y}^L$ for instance X^L. The solution $\hat{Y}^R$ for instance X^R is constructed similarly.

In order to construct the solution $\hat{Y}^L$, we start with the solution Y^L, and then gradually modify it over the course of $|\mathcal{R}^L|$ iterations, where in each iteration we reduce the number of points in the solution Y^L by at least 1, and we eliminate at most one row from $\mathcal{R}^L$. In order to execute an iteration, we select two rows R, R', with the following properties:

- Row R contains a point of X^L;
- Row R' contains a point of Y^L and it contains no points of X^L; and
- No point of $X^L \cup Y^L$ lies strictly between rows R and R'.

Note that, if $\mathcal{R}^L \neq \emptyset$, such a pair of rows must exist. We then collapse the row R' into the row R, obtaining a new modified solution to instance X^L (we use Observation 2.4). We claim that the number of points in the new solution decreases by at least 1. In order to show this, it is sufficient to show that there must be two points $p \in R$, $p' \in R'$ with the same x-coordinates; after the two rows are collapsed, these two points are mapped to the same point. Assume for contradiction that no such two points exist. Let $p \in R$, $p' \in R'$ be two points with smallest horizontal distance. Then it is easy to see that no point of $X^L \cup Y^L$ lies in the rectangle $\square_{p,p'}$, contradicting the fact that Y^L is a feasible solution for X^L.

In order to complete the proof of Claim 8.1, it is now enough to prove Observation 8.2.

Proof. Observation 8.2 We denote $X = \{p_1, \ldots, p_m\}$, where the points are indexed in the increasing order of their y-coordinates. Recall that a pair (p_i, p_{i+1}) of points is a crossing, if the two points lie on opposite sides of the line L. We say that it is a *left-to-right* crossing if p_i lies to the left of L, and we say that it is a *right-to-left* crossing otherwise. Clearly, either at least half the crossings of L are left-to-right crossings, or at least half the crossings of L are right-to-left crossings. We assume w.l.o.g. that it is the former. Let Π denote the set of all left-to-right crossings of L, so $|\Pi| \ge \mathsf{cost}(v)/2$. Notice that every point of X participates in at most one crossing in Π. We will associate, to each crossing $(p_i, p_{i+1}) \in \Pi$, a unique row in $\mathcal{R}^L \cup \mathcal{R}^R$. This will prove that $|\mathcal{R}^L| + |\mathcal{R}^R| \ge |\Pi| \ge \mathsf{cost}(v)/2$.

Consider now some crossing (p_i, p_{i+1}). Assume that p_i lies in row R, and that p_{i+1} lies in row R'. Let $\mathcal{R}_i$ be a set of all rows lying between R and R', including these two rows. We will show that at least one row of $\mathcal{R}_i$ lies in $\mathcal{R}^L \cup \mathcal{R}^R$. In order to do so, let H be the closed horizontal strip whose bottom and top boundaries are R and R', respectively. Let H^L be the area of H that lies to the left of the line L, and that excludes the row R – the row containing the point p_i, that also lies to the left of L. Similarly, let H^R be the area of H that lies to the right of the line L, and that excludes the row R'. Notice that, if any point $y \in Y^L$ lies in H^L, that the row containing y must belong to $\mathcal{R}^L$. Similarly, if any point $y' \in Y^R$ lies in H^R, then the row containing y' belongs to $\mathcal{R}^R$. Therefore, it is now sufficient to show that either H^L contains a point of Y^L, or H^R contains a point of Y^R. Assume for contradiction that this is false. Let $p \in X^L \cup Y^L$ be the point lying on the row R furthest to the right (such a point must exist because we can choose $p = p_i$). Similarly, let $p' \in X^R \cup Y^R$ be the point lying on the row R' furthest to the left (again, such a point must exist because we can choose $p' = p_{i+1}$.) But if H^L contains no points of Y^L, and H^R contains no points of Y^R, then no points of $X \cup Y$ lie in the rectangle $\square_{p,p'}$, and so the pair (p, p') of points is not satisfied in $X \cup Y$, a contradiction. □

□

8.2 Guillotine bound

In this section we prove Lemma 5.3, by showing that for any point set X that is a permutation, $\mathsf{GB}(X) \le 2\mathsf{OPT}(X)$. In order to do so, it is enough to prove that, for any point set X that is a permutation, for any partitioning tree T for X, $\mathsf{GB}_T(X) \le 2\mathsf{OPT}(X)$.

The proof is by induction on the height of T, and it is almost identical to the proof of Claim 2.7 for the standard Wilber Bound. When the height of the tree T is 1, then $|X| = 1$, so $\mathsf{GB}(X) = 0$ and $\mathsf{OPT}(X) = 0$.

Consider now a partitioning tree T whose height is greater than 1. Let T_1, T_2 be the two subtrees of T, obtained by deleting the root vertex r from T. Let (X_1, X_2) be the partition of X into two subsets given by the line $L(r)$, such that T_1 is a partitioning tree for X_1 and T_2 is a partitioning tree for X_2. Notice that, from the definition of the GB bound:

$$\mathsf{GB}_T(X) = \mathsf{GB}_{T_1}(X_1) + \mathsf{GB}_{T_2}(X_2) + \mathsf{cost}(r).$$

Moreover, from the induction hypothesis, $\mathsf{GB}_{T_1}(X_1) \le 2\mathsf{OPT}(X_1)$ and $\mathsf{GB}_{T_2}(X_2) \le 2\mathsf{OPT}(X_2)$. Using Claim 8.1 (that can be easily adapted to horizontal partitioning lines), we get that:

$$\mathsf{OPT}(X) \ge \mathsf{OPT}(X_1) + \mathsf{OPT}(X_2) + \mathsf{cost}(r)/2.$$

Therefore, altogether we get that:

$$\mathsf{GB}_T(X) \le 2\mathsf{OPT}(X_1) + 2\mathsf{OPT}(X_2) + \mathsf{cost}(r) \le 2\mathsf{OPT}(X).$$

References

[1] GEORGII MAKSIMOVICH ADEL'SON-VEL'SKII AND EVGENII MIKHAILOVICH LANDIS: An algorithm for organization of information. *Dokl. Akad. Nauk SSSR (Russian)*, 146(2):263–266, 1962. Math-Net.ru. 2

[2] RUDOLF BAYER: Symmetric binary B-trees: Data structure and maintenance algorithms. *Acta Informatica*, 1(4):290–306, 1972. [doi:10.1007/BF00289509] 2

[3] PROSENJIT BOSE, KARIM DOUÏEB, JOHN IACONO, AND STEFAN LANGERMAN: The power and limitations of static binary search trees with lazy finger. *Algorithmica*, 76:1264–1275, 2016. Preliminary version in ISAAC'14. [doi:10.1007/s00453-016-0224-x] 3

[4] PARINYA CHALERMSOOK, JULIA CHUZHOY, AND THATCHAPHOL SARANURAK: Pinning down the strong Wilber 1 bound for binary search trees. In *Proc. 23rd Internat. Conf. on Approximation Algorithms for Combinat. Opt. Probl. (APPROX'20)*, pp. 33:1–21. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020. [doi:10.4230/LIPIcs.APPROX/RANDOM.2020.33] 8

[5] PARINYA CHALERMSOOK, MAYANK GOSWAMI, LÁSZLÓ KOZMA, KURT MEHLHORN, AND THATCHAPHOL SARANURAK: Greedy is an almost optimal deque. In *Proc. 17th Symp. on Algorithms and Data Structures (WADS'15)*, pp. 152–165. Springer, 2015. [doi:10.1007/978-3-319-21840-3_13] 3

[6] PARINYA CHALERMSOOK, MAYANK GOSWAMI, LÁSZLÓ KOZMA, KURT MEHLHORN, AND THATCHAPHOL SARANURAK: Pattern-avoiding access in binary search trees. In *Proc. 56th FOCS*, pp. 410–423. IEEE Comp. Soc., 2015. [doi:10.1109/FOCS.2015.32, arXiv:1507.06953] 3, 7

[7] RANJAN CHAUDHURI AND HARTMUT F. HÖFT: Splaying a search tree in preorder takes linear time. *SIGACT News*, 24(2):88–93, 1993. [doi:10.1145/156063.156067] 3

[8] RICHARD COLE: On the Dynamic Finger Conjecture for splay trees. Part II: The proof. *SIAM J. Comput.*, 30(1):44–85, 2000. [doi:10.1137/S009753979732699X] 3

[9] RICHARD COLE, BUD MISHRA, JEANETTE SCHMIDT, AND ALAN SIEGEL: On the Dynamic Finger Conjecture for splay trees. Part I: Splay sorting $\log n$-block sequences. *SIAM J. Comput.*, 30(1):1–43, 2000. [doi:10.1137/S0097539797326988] 3

[10] THOMAS H. CORMEN, CHARLES E. LEISERSON, RONALD L. RIVEST, AND CLIFFORD STEIN: *Introduction to Algorithms*. MIT press, 2022. MIT Press. 2

[11] ERIK D. DEMAINE, DION HARMON, JOHN IACONO, DANIEL M. KANE, AND MIHAI PĂTRAŞCU: The geometry of binary search trees. In *Proc. 20th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA'09)*, pp. 496–505. SIAM, 2009. ACM DL. 3, 4, 5, 9, 12, 36

[12] ERIK D. DEMAINE, DION HARMON, JOHN IACONO, AND MIHAI PĂTRAŞCU: Dynamic optimality–almost. *SIAM J. Comput.*, 37(1):240–251, 2007. Preliminary version in FOCS'04. [doi:10.1137/S0097539705447347] 3, 4, 6, 8, 12

[13] JONATHAN C. DERRYBERRY AND DANIEL DOMINIC SLEATOR: Skip-splay: Toward achieving the unified bound in the BST model. In *Proc. 11th Symp. on Algorithms and Data Structures (WADS'09)*, pp. 194–205. Springer, 2009. [doi:10.1007/978-3-642-03367-4_18] 3

[14] JONATHAN C. DERRYBERRY, DANIEL DOMINIC SLEATOR, AND CHENGWEN CHRIS WANG: A lower bound framework for binary search trees with rotations. Technical report, CMU-CS-05-187, 2005. Available on author's website. 3

[15] AMR ELMASRY: On the sequential access theorem and deque conjecture for splay trees. *Theoret. Comput. Sci.*, 314(3):459–466, 2004. [doi:10.1016/j.tcs.2004.01.019] 3

[16] GEORGE F. GEORGAKOPOULOS: Chain-splay trees, or, how to achieve and prove $\log\log N$-competitiveness by splaying. *Inform. Process. Lett.*, 106(1):37–43, 2008. [doi:10.1016/j.ipl.2007.10.001] 3, 8

[17] DION HARMON: *New Bounds on Optimal Binary Search Trees*. Ph. D. thesis, MIT, 2006. MIT. 3

[18] JOHN IACONO: In pursuit of the dynamic optimality conjecture. In A. BRODNIK, A. LÓPEZ-ORTIZ, V. RAMAN, AND A. VIOLA, editors, *Space-Efficient Data Structures, Streams, and Algorithms*, volume 8066 of *LNCS*, pp. 236–250. Springer, 2013. [doi:10.1007/978-3-642-40273-9_16] 4, 6, 12, 26, 36

[19] JOHN IACONO AND STEFAN LANGERMAN: Weighted dynamic finger in binary search trees. In *Proc. 27th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA'16)*, pp. 672–691. SIAM, 2016. [doi:10.1137/1.9781611974331.ch49] 3

[20] LÁSZLÓ KOZMA: *Binary search trees, rectangles and patterns*. Ph. D. thesis, Saarland University, Saarbrücken, Germany, 2016. Saarland U. 4, 6

[21] VICTOR LECOMTE AND OMRI WEINSTEIN: Settling the relationship between Wilber's bounds for dynamic optimality. In *Proc. 28th Eur. Symp. Algorithms (ESA'20)*, pp. 68:1–21. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020. [doi:10.4230/LIPIcs.ESA.2020.68, arXiv:1912.02858] 4, 5, 7

[22] CALEB C. LEVY AND ROBERT E. TARJAN: A new path from splay to dynamic optimality. In *Proc. 30th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA'19)*, pp. 1311–1330. SIAM, 2019. [doi:10.1137/1.9781611975482.80] 37

[23] JOAN M. LUCAS: On the competitiveness of splay trees: Relations to the union-find problem. In LYLE A. MCGEOCH AND DANIEL D. SLEATOR, editors, *On-line Algorithms*, volume 7 of *DIMACS Ser. in Discrete Math. and Theor. Comp. Sci.*, pp. 95–124. Amer. Math. Soc., 1992. [doi:10.1090/dimacs/007] 3

[24] SETH PETTIE: Splay trees, Davenport-Schinzel sequences, and the Deque Conjecture. In *Proc. 19th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA'08)*, pp. 1115–1124. SIAM, 2008. [doi:10.5555/1347082.1347204, arXiv:0707.2160] 3

[25] DANIEL DOMINIC SLEATOR AND ROBERT ENDRE TARJAN: Self-adjusting binary search trees. *J. ACM*, 32(3):652–686, 1985. Preliminary version in STOC'83. [doi:10.1145/3828.3835] 3

[26] RAJAMANI SUNDAR: On the Deque Conjecture for the splay algorithm. *Combinatorica*, 12(1):95–124, 1992. [doi:10.1007/BF01191208] 3

[27] ROBERT ENDRE TARJAN: Sequential access in splay trees takes linear time. *Combinatorica*, 5(4):367–378, 1985. [doi:10.1007/BF02579253] 3

[28] CHENGWEN C. WANG, JONATHAN C. DERRYBERRY, AND DANIEL DOMINIC SLEATOR: O(log log N)-competitive dynamic binary search trees. In *Proc. 17th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA'06)*, pp. 374–383. SIAM, 2006. ACM DL. 3, 4, 6, 8

[29] ROBERT E. WILBER: Lower bounds for accessing binary search trees with rotations. *SIAM J. Comput.*, 18(1):56–67, 1989. Preliminary version in FOCS'86. [doi:10.1137/0218004] 3, 6, 12, 25, 26, 37, 65

AUTHORS

Parinya Chalermsook
Associate professor
Department of Computer Science
Aalto University
Espoo, Finland
chalermsook@gmail.com
`https://sites.google.com/site/parinyachalermsook/`

Julia Chuzhoy
Professor
Toyota Technological Institute at Chicago
Chicago, IL, USA
cjulia@ttic.edu
`https://home.ttic.edu/~cjulia/`

Thatchaphol Saranurak
Assistant professor
University of Michigan
Ann Arbor, MI, USA
thsa@umich.edu
https://sites.google.com/site/thsaranurak/

ABOUT THE AUTHORS

PARINYA CHALERMSOOK is a faculty member at the Department of Computer Science, Aalto University (Finland). He completed his Ph. D. at The University of Chicago under the supervision of Julia Chuzhoy and Janos Simon. He was at Max Planck Institute for Informatics as a postdoc and a senior research scientist from 2013 to 2016. Parinya is broadly interested in algorithms and extremal combinatorics. When he is not doing mathematics, he enjoys reading about political philosophy.

JULIA CHUZHOY is a Professor at the Toyota Technological Institute at Chicago. She completed her Ph. D. in Technion, Israel, and spent three years as a postdoctoral scholar at MIT, the University of Pennsylvania and the Institute for Advanced Study in Princeton. She mainly works on algorithms for graph problems. In her spare time she likes to read books, learns to play piano and studies French.

THATCHAPHOL SARANURAK is a faculty member of the Electrical Engineering and Computer Science Department at the University of Michigan. Prior to this, he spent two years as a research assistant professor at the Toyota Technological Institute at Chicago. Thatchaphol received his Ph. D. from KTH Royal Institute of Technology, Stockholm, in 2018 under the supervision of Danupon Nanongkai. His main research interest is in graph algorithms with a current focus on dynamic, local, and distributed algorithms. He likes sushi and Japanese manga.