Noname manuscript No.
(will be inserted by the editor)

An Asymptotically Optimal Algorithm for Online Stacking*

Martin Olsen · Lars Nørvang Andersen · Allan Gross

the date of receipt and acceptance should be inserted later

Abstract Consider a storage area where arriving items are stored temporarily in bounded capacity stacks until their departure. We look into the problem of deciding where to put an arriving item with the objective of minimizing the maximum number of stacks used over time. The decision has to be made as soon as an item arrives, and we assume that we only have information on the departure times for the arriving item and the items currently at the storage area. We are only allowed to put an item on top of another item if the item below departs at a later time. We refer to this problem as online stacking. We assume that the storage time intervals are picked i.i.d. from $[0,1] \times [0,1]$ using an unknown distribution with a bounded probability density function. Under this mild condition, we present a simple polynomial time online algorithm and show that the competitive ratio converges to 1 in probability. The result holds if the stack capacity is $o(\sqrt{n})$, where n is the number of items, including the realistic case where the capacity is a constant. Our experiments show that our results also have practical relevance.

Keywords Online algorithms · Stacking · Stowage · Asymptotic optimality

* A preliminary version of this paper appeared in the proceedings of the 6th International Conference on Computational Logistics, ICCL '15, under the title "Probabilistic Analysis of Online Stacking Algorithms" [13].

M. Olsen · A. Gross
Department of Business Development and Technology
Aarhus University
Denmark
E-mail: martino@btech.au.dk, agr@btech.au.dk

L. N. Andersen
Department of Mathematics
Aarhus University
Denmark
E-mail: larsa@math.au.dk

1 Introduction

In this paper, we consider the situation that some items arrive at a storage location where they are temporarily stored in LIFO stacks until their departure. When an item arrives, we are faced with the problem of deciding where to store the item. We will refer to this problem as the stacking problem. The stacking problem has many applications within real-world logistics. As an example, the items could be containers, and the storage location could be a container terminal or a container ship [4]. The items could also be steel bars [16] and trains [7], or the storage location could simply be a warehouse storing any type of objects stacked on top of each other.

We focus on the variant of the stacking problem given by the following assumptions: 1) We have to make a decision on where to store an item as soon as it arrives. When an item i arrives at time x_i, we are informed on the departure time y_i of the item, but we have no information on future items. In other words, we look at an *online* version of the problem, and we look for online algorithms solving the problem. 2) The numbers x_i and y_i could be any real numbers. This means that we restrict our attention to what we will refer to as the *continuous* case as opposed to the *discrete* case, where we only have a few possibilities for x_i and y_i. 3) We are only allowed to put an item i on top of an item j if $y_i \leq y_j$. Another way of saying this is that we do not allow *rehandling/relocations/overstowage* of items. 4) The objective is to minimize the maximum number of stacks in use over time given a bound h on the stacking height.

1.1 Contribution

We use the *unknown distribution model* for generating stacking problem instances, where the time intervals for storing the items are picked i.i.d. using an unknown distribution with bounded density:

Definition 1 The Unknown Distribution Model: Let n pairs $(a, b) \in [0, 1] \times [0, 1]$ be drawn i.i.d. using an unknown distribution with a bounded probability density function. For each pair (a, b), let an item arrive at the storage area at time $x = \min(a, b)$ and leave the storage area at time $y = \max(a, b)$.

If the reader prefers a model satisfying $a < b$, we can use a density f with $f(a, b) = 0$ for $a \geq b$. It is very common to use distributions with bounded densities to model real scenarios. For the univariate case, some examples of such distributions are uniform distributions, triangular distributions, and beta distributions with parameters greater than or equal to one. Assuming independence seems to be reasonable when items arrive at the storage area from different sources. This shows that our model is applicable for many realistic scenarios.

The main contribution of our paper is a simple polynomial time online algorithm called, for the lack of a better name, “Algorithm Online” for which

the following theorem holds, where $\chi_h \equiv \chi_{h,n}$ (suppressing the dependence on n in our notation) denotes the optimal number of stacks and $\chi'_h \equiv \chi'_{h,n}$ denotes the number of stacks used by the algorithm (the stacking problem is formally defined in Definition 2 in Sec. 2.1).

Theorem 1 *For the unknown distribution model, Algorithm Online produces a solution for the online stacking problem such that*

$$\lim_{n\to\infty} P\left(1 \leq \frac{\chi'_h}{\chi_h} \leq 1 + O(hn^{-\frac{1}{2}})\right) = 1 \ . \tag{1}$$

Algorithm Online processes one item in $O(\log n)$ time.

The following corollary is immediate:

Corollary 1 *If the stack capacity satisfies $h = o(\sqrt{n})$ (including the realistic case where h is a fixed constant) then the competitive ratio of Algorithm Online converges to 1 in probability:*

$$\frac{\chi'_h}{\chi_h} \xrightarrow{p} 1 \text{ for } n \to \infty \ .$$

Furthermore, if h is constant we will argue later that the competitive ratio is bounded and we have the following:

Corollary 2 *If h is a constant, then the expected value of the competitive ratio for Algorithm Online converges to 1 in the standard sense of convergence:*

$$E\left(\frac{\chi'_h}{\chi_h}\right) \to 1 \text{ for } n \to \infty \ .$$

To the best of our knowledge, we are the first to present an asymptotically optimal polynomial time online algorithm for stacking – an offline version has not been presented either. Algorithm Online has not been been presented earlier in the literature but similar algorithms have appeared [4,8,9,18]. No formal proof of asymptotic optimality for an online stacking algorithm has appeared earlier, so the most important part of the contribution is the formal proof of asymptotic optimality under mild conditions.

We also verify the results experimentally using two types of distributions and instances with $2000 \leq n \leq 200000$ and $h = 5$. For all our instances, $\frac{\chi'_h}{\chi_h} \leq 1 + kn^{-\frac{1}{2}}$ for a moderate constant k depending on the distribution involved, indicating that our results also have practical importance. The biggest k values were around 30 for the distributions in our experiments.

1.2 Related Work

A preliminary version of this paper [13] was presented at the conference ICCL 2015. The results in the present version are more generic and stronger since they are based on the unknown distribution model as compared to the results obtained in the preliminary version, which were based on a uniform distribution on the input. The present version furthermore includes a section with experiments.

The offline variant of the stacking problem where all information is provided before any decisions are made is NP-hard for any fixed bound $h \geq 6$ on the stacking height [5] as can be seen by reduction from the coloring problem on permutation graphs [10]. To the best of our knowledge, the computational complexity for the case $2 \leq h \leq 5$ is an open problem for the offline case. This variant of the problem is also NP-hard in the unbounded case as shown by Avriel et al. [2]. Tierney et al. [17] show that the problem of deciding if it is possible to accommodate all the items in a *fixed* number of bounded capacity stacks without relocations can be solved in polynomial time, but the running time of their offline algorithm is huge even for a small (fixed) number of stacks.

Cornelsen and Di Stefano [5] and Demange et al. [7] consider the problem in the context of assigning tracks to trains arriving at a train station/depot. Cornelsen and Di Stefano look at unbounded capacity stacks (train tracks) whereas Demange et al. consider unbounded as well as bounded capacity stacks. For unbounded stack capacity, Demange et al. show that no online stacking algorithm has a constant competitive ratio. In addition, they present lower and upper bounds for the competitive ratio with some improvements added later by Demange and Olsen [6]. For bounded capacity stacks, Demange et al. [7] show that $2 - 1/\min(h, \chi_h)$ is a tight bound for the competitive ratio for online stacking restricted to the situation where all trains are at the train depot at some point in time. This condition is known as *the midnight condition*. It is well-known that the stacking problem can be be solved exactly and online in polynomial time for the unbounded stack capacity case with the midnight condition by using the Patience Sorting method presented later in this paper.

Simple heuristics for online stacking similar to Algorithm Online have been presented by Borgman et al. [4], Duinkerken et al. [8], Hamdi et al. [9], and Wang et al. [18] without providing a proof of asymptotic optimality. Olsen shows in [12] how Reinforcement Learning can be used to improve simple online stacking heuristics. Finally, we mention the work of Rei and Pedroso [16] and König et al. [11] on related problems within the steel industry as well as the PhD thesis by Pacino [14] on container ship stowage.

1.3 Outline of the Paper

In Section 2, we look at the link between stacking problems and the coloring problems for overlap graphs and interval graphs and introduce some terminology used in this paper. We also consider some results from the field of

probability theory that form the basis for the probabilistic analysis of our online algorithm. Our algorithm is introduced in an offline and an online version in Section 3. The analysis of the algorithm and our main result are presented in Section 4, and finally, we verify our results experimentally in Section 5.

2 Preliminaries

In this section, we present the terminology used in this paper and some results from probability theory, which we will use later.

2.1 Connections to Graph Coloring

For each item i, we have an interval $I_i = [x_i, y_i]$ specifying the time interval for which the item has to be temporarily stored. To make it easier to formulate the constraint on the stacking height, we assume realistically that items cannot arrive and depart at exactly the same time. This assumption is consistent with the unknown distribution model that generates storage time intervals having pairwise distinct endpoints with probability 1.

It is well-known that the problem we consider can be formulated as a graph coloring problem [2], and we will use graph coloring terminology in the remaining part of the paper in order to make the presentation generic. We say that two intervals $I_1 = [x_1, y_1]$ and $I_2 = [x_2, y_2]$ *overlap* if and only if $x_1 < x_2 < y_1 < y_2$ or $x_2 < x_1 < y_2 < y_1$. We can put an item on top of another item if and only if their corresponding intervals do not overlap. Our problem can now be formally defined as follows, where h is the maximum allowed stack height:

Definition 2 The h-OVERLAP-COLORING problem:

- Instance: A set of n intervals $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$, where all the endpoints of the intervals are distinct.
- Solution: A coloring of the intervals using a minimum number of colors such that the following two conditions hold:
  1. Any two overlapping intervals have different colors.
  2. For any real number t and any color d, there will be no more than h intervals with color d that contain t.

It should be stressed that we look for online algorithms that process the intervals in order of increasing starting points.

The problem can be viewed as a graph coloring problem for the graph with a vertex for each interval and an edge between any two vertices where the corresponding intervals overlap. Such a graph is known as an *overlap graph*. As mentioned earlier, we let χ_h denote the minimum number of colors for a solution.

An *interval graph* is a graph in which each vertex corresponds to an interval and with an edge between two vertices if and only if the corresponding intervals

intersect. Note that two overlapping intervals intersect, but since the converse is not necessarily true, the overlap graph will be a subgraph of the interval graph. The classical greedy algorithm for graph coloring colors the vertices sequentially and works as follows: If we can reuse a color, we do so – otherwise we pick a new color that we have not used previously. The clique number of a graph is the size of a maximum clique. It is well-known and quite obvious that the greedy algorithm obtains a coloring with a number of colors corresponding to the clique number when it processes an interval graph in increasing order of the starting points of the intervals. It is also easy to see that we cannot obtain a better coloring than this.

2.2 Increasing Subsequences and Patience Sorting

The algorithm we present in Section 3 and the probabilistic analysis performed in Section 4 are based on some results from the theory on increasing subsequences and the method of Patience Sorting, which we will introduce next. Patience Sorting [1] is a method originally invented for sorting a deck of cards. Imagine that we have a deck of cards as follows, where the top of the deck is the leftmost card (the underlined cards will be explained later):

$$9, \underline{2}, \underline{4}, 8, 1, 7, \underline{6}, 3, 5, \underline{10}$$

We take the top card 9 and start a new pile. We now remove the other cards from the initial deck one by one from the top of the deck. Each time we remove a card, we try to put it in another pile with a top card of higher value than the removed card. If possible, we choose a pile where the top card has the lowest value. If not, we start a new pile. Card 2 goes on top of card 9 but we have to start two new piles with cards 4 and 8, respectively. Card 1 can be put on top of card 2, etc. Finally, we face the following four piles:

$$1, 2, 9 \qquad 3, 4 \qquad 5, 6, 7, 8 \qquad 10$$

It is now easy to sort the cards by repeatedly picking the top card with the lowest value. This is the Patience Sorting method, and we refer the reader to the work by Aldous [1] for more details.

Let L_n be the random variable representing the resulting number of piles for the Patience Sorting method on a deck with n cards. It is worth noting that L_n is identical to the length of the longest increasing subsequence for the sequence of cards defined by the deck. To illustrate this, there are several increasing subsequences that have length 4 for the sequence shown above (for example, the subsequence 2, 4, 6, 10, which is underlined) but no increasing subsequence with length 5 or more – and the number of piles needed is 4. Each pile represents a decreasing subsequence, and L_n is also the minimum number of decreasing subsequences into which the sequence can be partitioned. The asymptotic behavior of L_n has been extensively studied and we will need the

following result (eqn. (1.8) in [3]): For $M > 0$ sufficiently large there exists $d > 0$ and $D(M) \in \mathbb{R}$ such that when $M \leq t$

$$P(L_n > tn^{1/6} + 2\sqrt{n}) \leq D(M)\mathrm{e}^{-\mathrm{d}t^{3/5}} \tag{2}$$

Letting $t = n^{1/3}$ in (2), we see that there exists constants $D = D(M) \in \mathbb{R}$ and $d > 0$ such for that n sufficiently large it holds that

$$P(L_n > 3\sqrt{n}) \leq D\mathrm{e}^{-\mathrm{d}n^{1/5}} \tag{3}$$

For further results on L_n, we refer to [1,15].

Before we present our stacking strategy, we need to introduce a little more terminology. A *chain* of intervals is a sequence of intervals $I_1 \supseteq I_2 \supseteq I_3 \supseteq \ldots \supseteq I_m$. The intervals in a chain represent items that may be stacked on top of each other. We refer to the intervals I_1 and I_m as the *bottom* and the *top* of the chain, respectively. For a given number h, we can split a chain into chains of cardinality h or less in a natural way: The intervals I_1 to I_h form the first chain, the next h intervals I_{h+1} to I_{2h} form the next chain, etc. A partition of $\mathcal{I}$ into chains is a set of chains such that each interval is a member of exactly one chain.

3 The Algorithm

We present an offline and an online version of our algorithm named Offline and Online, respectively, which produce the same coloring for any instance of the h-OVERLAP-COLORING problem. Algorithm Offline is presented in order to make it easier for the reader to understand the coloring strategy used.

We are now ready to describe Algorithm Offline, which consists of 4 steps listed in Fig. 1. In the first step, we partition $\mathcal{I}$ into a minimum number c of chains (the number c can be determined using Patience Sorting as described in Lemma 1). This is illustrated in Fig. 2. In the second step, we split the chains into chains of cardinality h or less as described above. The interval graph of the bottoms of the chains is colored in the third step using the simple algorithm described in Section 2.1. Finally, in the fourth step, all the remaining intervals are colored with the color at the bottom of their chain. Steps 2, 3, and 4 are illustrated in Fig. 3 for the case $h = 2$. It is not hard to see that the coloring produced satisfies the conditions from Definition 2: All the chains produced in step 2 have cardinality at most h, and chain bottoms with the same color do not intersect.

We now prove that it is possible to transform Algorithm Offline into an online version, Algorithm Online, which is listed in Fig. 4. One step of Algorithm Online is also illustrated graphically in Fig. 5.

Lemma 1 *Algorithm Online is an online algorithm for the h-OVERLAP-COLORING problem producing a coloring identical to the coloring produced by Algorithm Offline. Algorithm Online processes one interval in* $O(\log n)$ *time.*

Algorithm Offline($\mathcal{I}$, h):
Step 1: Partition $\mathcal{I}$ into a minimum number c of chains.
Step 2: Split the chains into chains of cardinality h or less.
Step 3: Color the interval graph formed by the bottoms of the chains using the classical greedy algorithm.
Step 4: Color any interval not at the bottom of a chain with the color of the bottom of its chain.

Fig. 1: The offline version of our algorithm.

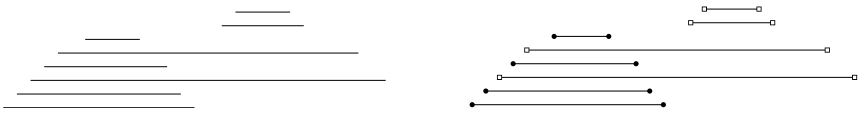

Fig. 2: The initial phase of Algorithm Offline is illustrated here. To the left, we see the intervals forming the instance. The two chains created in step 1 are shown to the right (the endpoints of the intervals are different for the two chains).

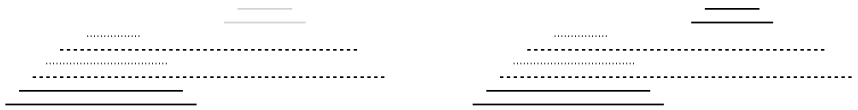

Fig. 3: This figure illustrates the final phase of Algorithm Offline for the case $h = 2$. The four chains produced in step 2 are shown to the left, and the coloring produced in steps 3 and 4 is shown to the right. The algorithm generates a coloring using $\chi'_h = 3$ colors.

Proof Let π be a permutation of the integers from 1 to n such that $x_{\pi(i)} < x_{\pi(j)}$ for $i < j$. Now we consider the sequence where the i'th number is $y_{\pi(i)}$. There is a simple one-to-one correspondence between a decreasing subsequence of this sequence and a chain of intervals from the set $\mathcal{I}$: If we start at the bottom of a chain and move upward, then the x-values increase and the y-values decrease. This means that we obtain a partition of $\mathcal{I}$ into a minimum number of chains, if we apply the Patience Sorting method described in Section 2.2 and partition the sequence into a minimum number of decreasing subsequences.

Algorithm Online processes the intervals in increasing order of their starting points applying the Patience Sorting method, and decisions on an interval are made without considering intervals with bigger starting points. The same goes for the splitting into smaller chains as well as the coloring of the chain bottoms and the other intervals. This means that Algorithm Online is an online algorithm producing the same coloring as Algorithm Offline.

Each step of the Patience Sorting method requires $O(\log n)$ time if we use binary search to locate the right pile. Keeping track of unused colors can also be handled in $O(\log n)$ time for each step if a priority queue is used (a priority

Algorithm Online($\mathcal{I}$, h):
Assumption on $\mathcal{I} = \{[x_1, y_1], [x_2, y_2], \ldots, [x_n, y_n]\}$: $i < j \Rightarrow x_i < x_j$

```
 1: C ← ∅
 2: D ← ∅
 3: χ ← 0
 4: for i ∈ {1, 2, ..., n} do
 5:     bottom ← false
 6:     Let H be the set of chains in C where
        the top of the chain contains I_i.
 7:     if H = ∅ then
 8:         Add a new chain to C consisting of I_i.
 9:         bottom ← true
10:     else
11:         Let c_J be the chain in H with a top interval
            J[x_J, y_J] with the smallest value of y_J.
12:         Put I_i on top of c_J.
13:         Let d be the color assigned to J.
14:         if there are less than h intervals in c_J with color d then
15:             Assign color d to I_i.
16:         else
17:             bottom ← true
18:         end if
19:     end if
20:
21:     if bottom = true then
22:         Let G = {(d, y) ∈ D : y < x_i}
23:         if G = ∅ then
24:             χ ← χ + 1
25:             Assign color χ to I_i.
26:             D ← D ∪ {(χ, y_i)}
27:         else
28:             Pick any (d, y) ∈ G.
29:             Assign color d to I_i.
30:             D ← (D \ {(d, y)}) ∪ {(d, y_i)}
31:         end if
32:     end if
33: end for
```

Fig. 4: The online version of our algorithm. Please note that we assume the intervals in $\mathcal{I}$ to appear in increasing order of their starting points.

queue storing information on when the colors expire is used for the set $\mathcal{D}$ in Fig. 4). □

4 Probabilistic Analysis

Let ω' be the clique number of the interval graph formed by the set of intervals $\mathcal{I}$. In this section, we conduct the probabilistic analysis on the competitive ratio. First, however, in Lemma 2 and its associated corollary, we prove a non-random bound on the competitive ratio in terms of c (the minimum number of chains formed in step 1 of Algorithm Offline) and ω'.

Fig. 5: One iteration of the for loop of Algorithm Online is illustrated here ($h = 2$). To the left a new interval arrives (at the top with no endpoint symbols). The set $\mathcal{H}$ consists of two chains. To the right we see the new interval joining the chain with black circles as endpoints. The new interval receives the same color as the interval below it in that chain.

Lemma 2 *The coloring produced by Algorithm Offline and Online uses χ'_h colors satisfying*

$$\chi'_h \leq \frac{\omega'}{h} + c \ . \tag{4}$$

Proof For any real number x, we let g_x denote the number of intervals in $\mathcal{I}$ that contain x and let g_x^h denote the number of chain bottoms produced in step 2 of Algorithm Offline containing x. As mentioned in Section 2.1, any interval graph can be colored with a number of colors corresponding to the size of the largest clique of the graph:

$$\chi'_h = \max_x g_x^h \ . \tag{5}$$

Now consider an interval that is a bottom of a chain produced in step 2 of Algorithm Offline but not a bottom of one of the chains produced in step 1. If such an interval contains a number x, then the $h-1$ intervals directly below it in the chain will also contain x. There are at least $g_x^h - c$ such intervals that contain x so we obtain the following inequality:

$$(g_x^h - c)h \leq g_x \ . \tag{6}$$

We now rearrange this inequality:

$$\max_x g_x^h \leq \frac{\max_x g_x}{h} + c \ . \tag{7}$$

Next, we use (5) and $\omega' = \max_x g_x$. □

At some point in time there are ω' items in our storage area implying $\chi_h \geq \frac{\omega'}{h}$. We now have the following corollary:

Corollary 3 *The competitive ratio of Algorithm Online satisfies*

$$1 \leq \frac{\chi'_h}{\chi_h} \leq 1 + h\frac{c}{\omega'} \ .$$

Going forward, we regard c and ω' as random and our aim is then to show that the competitive ratio $\frac{\chi'_h}{\chi_h}$ of Algorithm Online is close to 1 with high probability. Formally, we say that a sequence of events $\{E_n\}$ occurs *with high probability*, abbreviated whp, if $P(E_n) \to 1$ for $n \to \infty$. We will now show

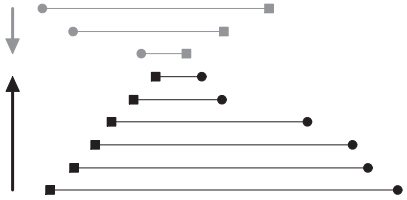

Fig. 6: The figure shows a decreasing subsequence for the sequence of b-values. The squares and circles correspond to a-values and b-values, respectively. The decreasing subsequence can be split into a grey chain and a black chain of intervals.

that the competitive ratio is $1+O(hn^{-\frac{1}{2}})$ whp. The strategy of our proof is to show that $c = O(\sqrt{n})$ whp and that $\omega' = \Omega(n)$ whp, and then combine these results.

For a brief moment, we leave the unknown distribution model and present a lemma for a simpler model for generating the instances: *the uniform model*. This model is obtained by substituting the unknown distribution in the unknown distribution model (see Definition 1) with the uniform distribution on $[0,1] \times [0,1]$. This is the only place in the paper where we are *not* using the unknown distribution model.

Lemma 3 *For the uniform model, the set of intervals $\mathcal{I}$ can be partitioned into c chains such that*

$$c \leq 6\sqrt{n} \text{ whp.}$$

Proof Let (a_i, b_i) denote the i'th pair drawn using the uniform model. We introduce a permutation π' on the integers from 1 to n defined by $a_{\pi'(i)} < a_{\pi'(j)}$ for $i < j$. We now look at the sequence of b-values with $b_{\pi'(i)}$ as the i'th number in the sequence. We use the Patience Sorting method from Section 2.2 on the b-sequence and obtain c' decreasing subsequences. We split each subsequence into two decreasing subsequences if there is a point where the b-values become lower than their corresponding a-values. It is not hard to see that we can form a chain of intervals for each of the up to $2c'$ subsequences we obtain by the splitting procedure (see Fig. 6). Since $c \leq 2c'$, we have the following:

$$P(c > 6\sqrt{n}) \leq P\left(c' > 3\sqrt{n}\right) \quad . \tag{8}$$

The a- and b-values are independent for the uniform model, so c' and L_n have the same distribution, where L_n is the length of the longest increasing subsequence for a permutation of n numbers chosen uniformly at random (see Section 2.2):

$$P\left(c' > 3\sqrt{n}\right) = P\left(L_n > 3\sqrt{n}\right) \quad . \tag{9}$$

We know from (3) that for n sufficiently large we have

$$P(L_n > 3\sqrt{n}) \leq D\mathrm{e}^{-\mathrm{d}n^{1/5}}$$

for constants $D \in \mathbb{R}$ and $d > 0$, and hence for n tending to infinity, we obtain

$$\lim_{n\to\infty} P(L_n > 3\sqrt{n}) \to 0,$$

and we see that combining (8) and (9) proves the lemma. □

We now use this lemma for the uniform model to prove a similar lemma for the more generic unknown distribution model.

Lemma 4 *For the unknown distribution model, the set of intervals $\mathcal{I}$ can be partitioned into c chains such that*

$$c = O(\sqrt{n}) \text{ whp.}$$

Proof Let $B > 1$ be such that $f(a,b) \leq B$ and let the function $g : [0,1] \times [0,1] \to \mathbb{R}_{\geq 0}$ be defined as follows:

$$g(a,b) = \frac{1 - f(a,b)/B}{1 - 1/B} \ .$$

The function g clearly qualifies as a probability density function. Consider the following procedure for sampling from $[0,1]^2$:

♠ Pick a pair (a,b) using f with probability $1/B$ or g with probability $1 - 1/B$.

If we let $\mathbf{U}$ and U be independent uniform distributions on $[0,1]^2$ and $[0,1]$, respectively, then g is the density of $\mathbf{U} \mid \{U > f(\mathbf{U})/B\}$, i.e. $\mathbf{U}$ conditioned on the event $A \stackrel{\text{def}}{=} \{U > f(\mathbf{U})/B\}$. Furthermore, since the density of $\mathbf{U} \mid \{U \leq f(\mathbf{U})/B\}$ is f and $P(A) = 1 - 1/B$, it follows from the law of total probability, that the procedure ♠ will yield a uniform distribution on $[0,1]^2$. This fact can also be verified from direct calculation, since $1/B \cdot f + (1 - 1/B) \cdot g = 1$.
The proof proceeds by considering the minimum number of chains when picking n intervals using f and relating this quantity to the minimum number of chains picking $\lfloor Bn \rfloor$ intervals using the ♠ procedure, so to distinguish between the two, we let $c \equiv c_{f,n}$ denote the former number and let $\tilde{c}$ denote the latter. Furthermore, we let F_n denote the number of intervals picked using the density f when using the ♠ procedure.
For $x \in \mathbb{R}$ consider the probability that n intervals drawn using the density f can be partitioned into more than x chains. The corresponding probability can only increase if we pick $\lfloor Bn \rfloor$ intervals using the procedure ♠ when it is given that n intervals have been drawn using the density f:

$$P(c > x) \leq P(\tilde{c} > x | F_n \geq n) \ . \tag{10}$$

Notice, that even though the intervals picked using the ♠ procedure are marginally uniform, this is not the case when we condition on F_n, so Lemma 3 is not directly applicable to (10). However, we have

$$P(\tilde{c} > x | F_n \geq n) = \frac{P(\tilde{c} > x, F_n \geq n)}{P(F_n \geq n)} \leq \frac{P(\tilde{c} > x)}{P(F_n \geq n)} \, ,$$

where we notice, that $\tilde{c}$ in the numerator above is based on intervals selected using the uniform model. Furthermore, since F_n has a binomial distribution: $F_n \sim B(\lfloor Bn \rfloor, 1/B)$ we have $\lim_n P(F_n \geq n) = 1/2$ by the Central Limit Theorem, so that if let $x = 6\sqrt{\lfloor Bn \rfloor}$ and apply Lemma 3, we obtain

$$\lim_n P(c > x) \leq \lim_n P(\tilde{c} > x | F_n \geq n) = 0$$

which gives the desired result. □

As a side remark, it should be noted that the upper bound in Lemma 4 matches and extends the result for the uniform distribution ($B = 1$) from Lemma 3.

To illustrate a case where the premises of Lemma 4 are *not* satisfied, we can for instance consider a model where the departure time is fixed at 0.1 after the arrival time which itself is uniform in $[0, 0.9]$, or, in other words where

$$(x, y) \mid u \sim (u, u + 0.1) \qquad \text{and} \quad u \sim U(0, 1)\,.$$

It is easy to see that $c = n$ with probability 1 in this case. Ironically, our algorithm works perfectly when intervals are picked using this stochastic process.

Recall that ω' is the clique number of the interval graph formed by the set of intervals $\mathcal{I}$.

Lemma 5 *For the unknown distribution model, we have the following:*

$$\omega' = \Omega(n) \text{ whp.}$$

Proof The triangle above the diagonal $y = x$ in the square $[0, 1] \times [0, 1]$ can be partitioned into squares S'_z as illustrated in Fig. 7. Here, $0 < z < 1$ is a *dyadic rational* i.e. of the form $z = k2^{-m}$ for integers k and m. The triangle below the diagonal can be partitioned in a similar way using squares S_z. We now have the following:

$$\iint_{[0,1]\times[0,1]} f(a, b)\, da\, db = \sum_z \iint_{S_z \cup S'_z} f(a, b)\, da\, db = 1\ . \tag{11}$$

At least one of the summands on the rhs. of (11) must be strictly greater than 0, i.e. there exists z_0 such that

$$\iint_{S_{z_0} \cup S'_{z_0}} f(a, b)\, da\, db > 0\ .$$

Let $K_{z_0} \stackrel{\text{def}}{=} \#\{i : z_0 \in I_i\}$ be the number of intervals containing z_0 and note that

$$\{\omega' \geq k\} \supseteq \{K_{z_0} \geq k\}$$

and since it holds for all i and all z that

$$P(z \in I_i) \geq \iint_{S_z \cup S'_z} f(a, b)\, da\, db\ ,$$

it follows from the weak law of large numbers, that z_0 will be contained in $\Omega(n)$ intervals whp. □

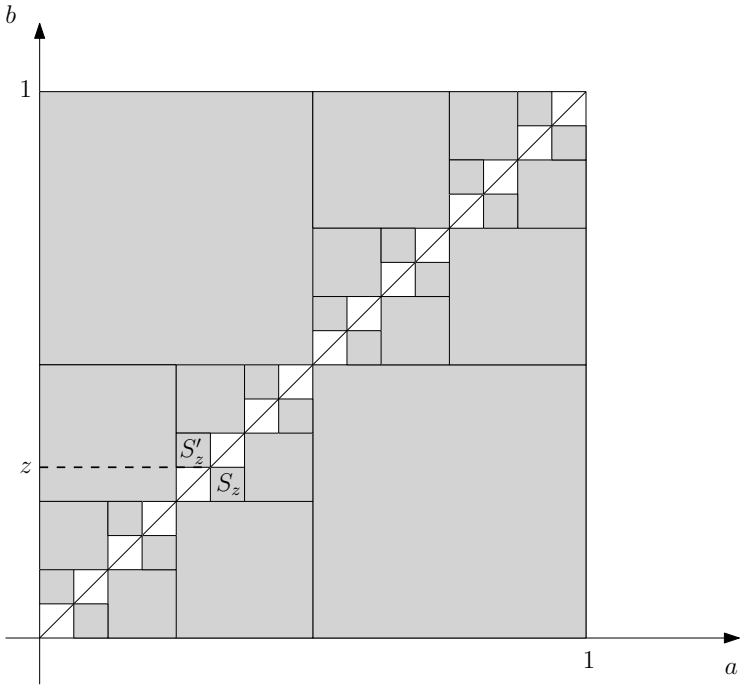

Fig. 7: The square $[0,1] \times [0,1]$ (except the diagonal $y = x$) partitioned into smaller squares S'_z and S_z.

We now present a proof of the main theorem of the paper:

Proof (Theorem 1) Algorithm Online is an online algorithm using $O(\log n)$ time per item according to Lemma 1. We can combine the results of Lemma 4 and Lemma 5 in the sense that if we define events $A_n \stackrel{\text{def}}{=} \{c \leq \alpha\sqrt{n}\}$ and $B_n = \{\omega' \geq \beta n\}$ where α and β are the implied constants from the O-notation, we have (using Boole's inequality)

$$P\left(\frac{c}{\omega'} \leq \frac{\alpha}{\beta}\frac{1}{\sqrt{n}}\right) \geq 1 - P(\overline{A_n}) - P(\overline{B_n})$$

and letting n tend to infinity, we see that $c/\omega' = O(1/\sqrt{n})$ whp which together with Corollary 3 concludes the proof. □

Corollary 1 is an immediate consequence of Theorem 1 and to see Corollary 2 we note that it is not hard to prove that $\chi'_h \leq \omega'$, which implies $\frac{\chi'_h}{\chi_h} \leq h$ i.e. the competitive ratio is bounded, and, as such, uniformly integrable which together with the convergence in probability from Corollary 1 yields the desired result. (In fact, we obtain convergence of the m-th moment, for any m.)

5 Experiments

We have performed some experiments to verify the theoretical results and to examine the underlying constants for the big O notation. For the first type of experiments, we have used the unknown distribution model introduced in Definition 1 with a uniform distribution on $\{(a, b) \in [0, 1] \times [0, 1] : |a - b| \leq \ell\}$ for some number ℓ as the "unknown" distribution. In other words, we are choosing an interval with length up to ℓ uniformly at random. We use the notation $U(\ell)$ for this type of experiment.

For the second type of experiments, we go beyond the unknown distribution model and choose the center and the length of an interval independently using two normal (Gaussian) distributions (if the length is negative, then we ignore it and pick a new length). This means that any real number can be an interval endpoint. The notation $N(\mu_c, \sigma_c^2, \mu_l, \sigma_l^2)$ is used for the second type of experiments, where μ_c and σ_c are the mean and the standard deviation for the center of an interval, and μ_l and σ_l are the corresponding entities for the length of an interval. We go beyond the unknown distribution model to look into an even broader setting.

The eight distributions that we have used are $U(\ell)$, $\ell \in \{0.1, 0.3, 0.5, 0.8\}$, and $N(\mu_c, \sigma_c^2, \mu_l, \sigma_l^2)$, $(\mu_c, \sigma_c, \mu_l, \sigma_l) \in \{(0, 1, 1, 0.2), (0, 1, 1, 0.4), (0, 5, 1, 0.2), (0, 5, 1, 0.4)\}$.

The stack capacity has been fixed to $h = 5$ for all the experiments. The experiments examine three perspectives corresponding to the three subsections in this section. For every combination of the eight distributions and three perspectives, we have generated 100 random instances: one instance for each n in the set $\{2000, 4000, 6000, 8000, 10000, \ldots, 200000\}$. Please note that no instances have been reused.

5.1 Experiments for the Number of Chains

Lemma 4 is a key lemma specifying an upper bound on c, i.e., the minimum number of chains that can be formed for an instance of the stacking problem. The values of $c/\sqrt{n}$ have been plotted against n in Fig. 8 for the two types of distributions that we consider.

The experiments clearly verify Lemma 4 by showing that $c/\sqrt{n} = O(1)$ – even when we go beyond our unknown distribution model using the Gaussian distributions. The underlying constant k seems to be moderate (not bigger than 15 for our distributions), and $c/\sqrt{n} \leq k$ holds for <u>all</u> the instances with k depending on the actual distribution.

5.2 Convergence Rate Experiments

We now take a closer experimental look at our main contribution: Theorem 1. Our purpose is to verify the theorem and examine the actual convergence rate

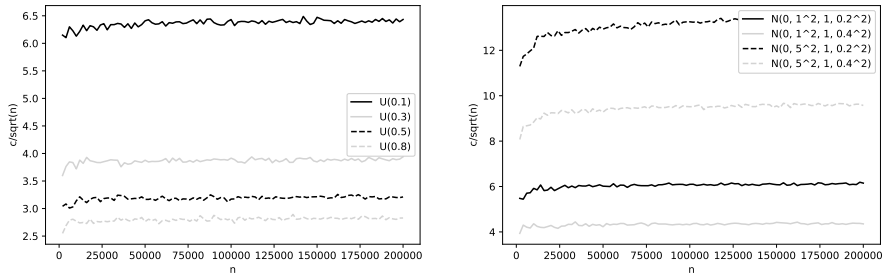

Fig. 8: The values of $c/\sqrt{n}$ plotted against n for the Uniform (left) and Gaussian type of distributions (right).

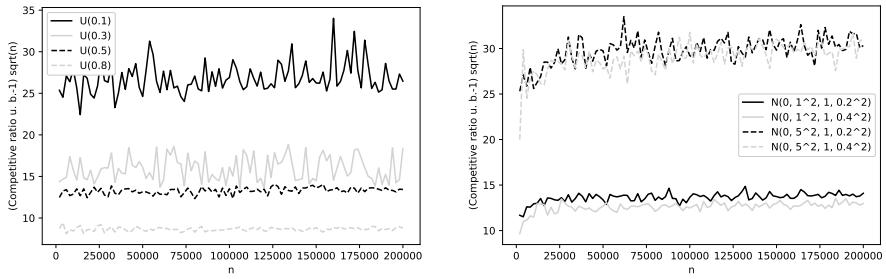

Fig. 9: The graph shows the results of the experiments for the expression $\left(\frac{\chi'_h}{(\omega'/h)} - 1\right)\sqrt{n}$ for the Uniform (left) and Gaussian type of distributions (right).

for the eight distributions that we consider. Directly inspired by our theorem, we have plotted $\left(\frac{\chi'_h}{(\omega'/h)} - 1\right)\sqrt{n}$ against n in Fig. 9. We remind the reader that $\chi_h \geq \frac{\omega'}{h}$ so $\frac{\chi'_h}{(\omega'/h)}$ is an upper bound on the competitive ratio that we can efficiently compute (as mentioned earlier, we have no efficient procedure for computing χ_h for $h = 5$ at the moment).

Similar to the experiments with the number of chains c, we conclude that $\left(\frac{\chi'_h}{(\omega'/h)} - 1\right)\sqrt{n} = O(1)$ with an underlying moderate constant k. The maximum k-values observed for our distributions were around 30. From the graphs, we can see that $\frac{\chi'_h}{(\omega'/h)} \leq 1 + k/\sqrt{n}$ is satisfied for all our instances.

5.3 Competitive Ratio Experiments

For the sake of completeness, we ran some experiments and plotted the upper bound for the competitive ratio, $\frac{\chi'_h}{(\omega'/h)}$, against n. The results are shown in Fig. 10.

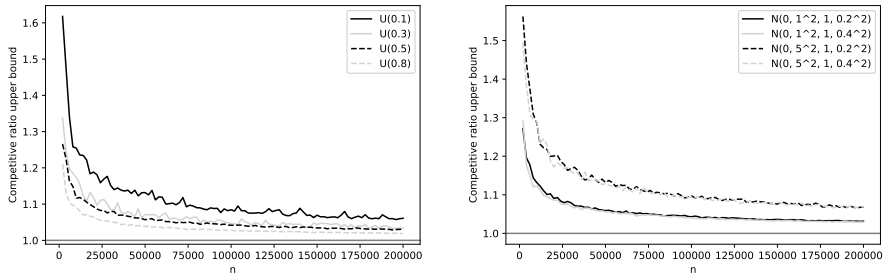

Fig. 10: An upper bound for the competitive ratio plotted against n for the Uniform (left) and Gaussian type of distributions (right).

These graphs confirm that the competitive ratio converges to 1 in probability. The competitive ratio is relatively close to 1 even for instances with not that many items. As an example, the competitive ratio is not bigger than 1.2 for instances with 25000 items for the distributions that we have considered.

6 Conclusion

We have presented a simple polynomial time online algorithm for stacking with a competitive ratio that converges to 1 in probability under the unknown distribution model.

The main message of our paper is that such an algorithm exists. The experimental part of our paper shows that the results also have practical relevance. We strongly believe that there are other asymptotically optimal algorithms for online stacking. For future research, we propose to formally analyze similar algorithms to check if they are asymptotically optimal. It is also important to compare the convergence rates for the competitive ratios for different asymptotically optimal algorithms if more than one algorithm exists.

Acknowledgements

The authors thank the anonymous reviewers for their valuable comments and suggestions.

References

1. Aldous, D., Diaconis, P.: Longest increasing subsequences: From patience sorting to the baik-deift-johansson theorem. Bull. Amer. Math. Soc **36**, 413–432 (1999)
2. Avriel, M., Penn, M., Shpirer, N.: Container ship stowage problem: complexity and connection to the coloring of circle graphs. Discrete Applied Mathematics **103**(1-3), 271 – 279 (2000)

3. Baik, J., Deift, P., Johansson, K.: On the distribution of the length of the longest increasing subsequence of random permutations. Journal of the American Mathematical Society **12**(4), 1119–1178 (1999). URL http://www.jstor.org/stable/2646100
4. Borgman, B., van Asperen, E., Dekker, R.: Online rules for container stacking. OR Spectrum **32**(3), 687–716 (2010)
5. Cornelsen, S., Stefano, G.D.: Track assignment. Journal of Discrete Algorithms **5**(2), 250 – 261 (2007)
6. Demange, M., Olsen, M.: A note on online colouring problems in overlap graphs and their complements. In: WALCOM 2018, *Lecture Notes in Computer Science*, vol. 10755, pp. 144–155. Springer (2018)
7. Demange, M., Stefano, G.D., Leroy-Beaulieu, B.: On the online track assignment problem. Discrete Applied Mathematics **160**(7-8), 1072–1093 (2012)
8. Duinkerken, M.B., Evers, J.J.M., Ottjes, J.A.: A simulation model for integrating quay transport and stacking policies on automated container terminals. In: Proceedings of the 15th European Simulation Multiconference (ESM2001) (2001)
9. Hamdi, S.E., Mabrouk, A., Bourdeaud'Huy, T.: A heuristic for the container stacking problem in automated maritime ports. IFAC Proceedings Volumes **45**(6), 357 – 363 (2012)
10. Jansen, K.: The mutual exclusion scheduling problem for permutation and comparability graphs. Information and Computation **180**(2), 71 – 81 (2003)
11. König, F.G., Lübbecke, M.E., Möhring, R.H., Schäfer, G., Spenke, I.: Solutions to real-world instances of pspace-complete stacking. In: Algorithms - ESA 2007: 15th Annual European Symposium, *Lecture Notes in Computer Science*, vol. 4698, pp. 729–740. Springer (2007)
12. Olsen, M.: Online stacking using RL with positional and tactical features. In: Learning and Intelligent Optimization - 14th International Conference, LION 14, *Lecture Notes in Computer Science*, vol. 12096, pp. 184–194. Springer (2020)
13. Olsen, M., Gross, A.: Probabilistic analysis of online stacking algorithms. In: Computational Logistics - 6th International Conference, ICCL 2015, *Lecture Notes in Computer Science*, vol. 9335, pp. 358–369. Springer (2015)
14. Pacino, D., Jensen, R.: Fast generation of container vessel stowage plans: using mixed integer programming for optimal master planning and constraint based local search for slot planning. Ph.D. thesis, IT University of Copenhagen (2012)
15. Pilpel, S.: Descending subsequences of random permutations. Journal of Combinatorial Theory, Series A **53**(1), 96 – 116 (1990)
16. Rei, R.J., Pedroso, J.P.: Tree search for the stacking problem. Annals OR **203**(1), 371–388 (2013)
17. Tierney, K., Pacino, D., Jensen, R.M.: On the complexity of container stowage planning problems. Discrete Applied Mathematics **169**(0), 225 – 230 (2014)
18. Wang, N., Zhang, Z., Lim, A.: The stowage stack minimization problem with zero rehandle constraint. In: IEA/AIE (2), *Lecture Notes in Computer Science*, vol. 8482, pp. 456–465. Springer (2014)