Online Algorithms for The Maximum k-Interval Coverage Problem

Songhua Li[1,2*], Minming Li[1*], Lingjie Duan[2*] and Victor C.S. Lee[3*]

[1]Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong SAR, China.
[2]Engineering Systems and Design Pillar, Singapore University of Technology and Design, Singapore, Singapore.
[3]Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong SAR, China.

*Corresponding author(s). E-mail(s):
songhuali3-c@my.cityu.edu.hk; minming.li@cityu.edu.hk;
lingjie_duan@sutd.edu.sg; csvlee@eee.hku.hk;

Abstract

We study the online maximum coverage problem on a target interval, in which, given an online sequence of sub-intervals (which may intersect among each other) to arrive, we aim to select at most $\boldsymbol{k}$ of the sub-intervals such that the total covered length of the target interval is maximized. The decision to accept or reject each sub-interval is made immediately and irrevocably right at the release time of the sub-interval. We comprehensively study various settings of this problem regarding both the length of each released sub-interval and the total number of released sub-intervals. To begin with, we investigate the offline version of the problem where the sequence of all the released sub-intervals is known in advance to the decision-maker and propose two polynomial-time optimal solutions to different settings of our offline problem. For the online problem, lower bounds on the competitive ratio are first proposed on our well-designed release schemes of sub-intervals. Then, we propose a Single-threshOld-based deterministic Algorithm (SOA), which adds a sub-interval if the added length without overlap exceeds a certain threshold, achieving competitive ratios close to the lower bounds. Further, we extend SOA to a Double threshOlds based deterministic Algorithm (DOA) by using the first threshold for exploration

and the second threshold (larger than the first one) for exploitation. With the two thresholds generated by our proposed program, we show that DOA outperforms SOA slightly in the worst-case scenario. Moreover, we show that more thresholds cannot induce better worst-case performance of an online deterministic algorithm as long as those thresholds are used in non-increasing order in accepting sub-intervals.

Keywords: Maximum k-Coverage Problem, Online Budgeted Maximum Coverage Problem, Interval Coverage, Online Algorithm

Acknowledgments

A preliminary version of this paper appeared in the proceedings of COCOA 2020 Li et al (2020). Minming Li is also from City University of Hong Kong Shenzhen Research Institute, Shenzhen, P.R. China.

1 Introduction

The classical Maximum k-Coverage Problem is a fundamental problem in computer science with a wide range of applications in different domains. The goal of this problem is to select k sets from a collection $\mathsf{S} = \{S_1, \cdots, S_n\}$ of subsets of a universal set $\mathsf{U} = \{u_1, \cdots, u_m\}$ where each element is associated with a weight $w : \mathsf{U} \to \mathbb{R}$. This problem is NP-hard and admits a $(1 - \frac{1}{e})$-approximation algorithm by greedily selecting a set that maximally increases the current overall coverage Hochbaum and Pathria (1998). When each $S_i \in \mathsf{S}$ is further associated with a cost $c : \mathsf{S} \to \mathbb{R}$ and the budget k is relaxed from an integer to a real number, we have the generalized Budgeted Maximum Coverage Problem (BMC) which targets at selecting a sub-collection of sets in S maximizing the total weight of the covered elements in U while adhering to the budget k. BMC is also NP-hard and has a $(1-\frac{1}{e})$-approximation algorithm Khuller et al (1999). In the classical online version of the maximum coverage problem, the system sequentially releases a set $S_i \in \mathsf{S}$ of elements together with their associated values at each timestamp i while an online algorithm must decide whether to accept or reject S_i at the same timestamp. Note that the classical online coverage problem usually allows an algorithm to irrevocably drop previously accepted sets Rawitz and Rosén (2021), which is not the case in our problem. Each rejected or dropped set cannot be retrieved at a later timestamp.

This paper studies the novel online maximum k-coverage problem on a line where any previously-accepted set (i.e., sub-interval) cannot be dropped, which we refer to as the Online Maximum k-Interval Coverage Problem as formally defined later in Section 2. In the problem, our goal is to accept $k \in \mathbb{N}$ sub-intervals from an online sequence of sub-intervals (which may intersect among each other) of a target interval such that the overall covered length

of the target interval is maximized. Note that the decision to either accept or reject each sub-interval should be made irrevocably at the release time of the sub-interval, and an accepted sub-interval cannot be dropped later. Apart from its nature in combinatorial optimization, our problem has a practical motivation in the context of crowd-sourcing activities where individuals can collectively contribute their expertise to a common goal. For example, the Amazon Mechanical Turk (mturk.com) recruits online workers to complete an MTurk task Brawley and Pury (2016). Facing a sequence of online workers, the crowd-sourcing decision-maker has a quota k of workers to hire and wants to diversify the k workers' domain knowledge (or, say, to maximize the final k workers' expertise width without counting the overlap). The decision for hiring each worker is usually irrevocable and must be made on arrival without knowing future workers' expertise.

Comprehensive settings of the problem are investigated in this paper. Regarding the overall number of sub-intervals (i.e., online workers) released, we consider the Fixed-Number (FN) setting where the number of online workers is known as a constant and the Variable-Number (VN) setting where the number of online workers is not known in advance. With regard to the length of a sub-interval (i.e., the width of an online worker's expertise range), we consider the Unit-Length (UL) setting where an online worker has a normalized unit expertise width, the Flexible-Length (FL) setting where the expertise width of an online worker varies in a fixed range and the Arbitrary-Length (AL) setting where the expertise width of an online worker varies arbitrarily, respectively. In addition, we extend UL to the Unit-Sum (US) setting, where an online worker has diverse expertise with sum width normalized as a unit. In a special case of our problem that any two sub-intervals are disjoint, each sub-interval could contribute its whole length to whichever solution it follows. As such, our problem is closely related to the k-secretary problem, which seeks to select k out of n online values such that the expected sum of selected values is maximized.

Note that our offline problem, where complete information of all sub-intervals to be released is given at the very beginning, is closely related to the interval maximum coverage problem as studied by Cohen and Gonen (2019). Given a ground set $G = \{g_1, ..., g_p\}$ of p real numbers with weights $\{w_1, ..., w_p\}$ and a set $\mathbb{V}_n = \{V_1, ..., V_n\}$ of n real intervals, they ask for a subset $U \in \mathbb{V}_n$ with size exactly equals k such that $\sum_{i=1}^{p} w_i \delta_i$ is maximized, in which $\delta_i = 1$ if $\bigcup_{V \in U} V$ covers element g_i. They propose a polynomial-time dynamic programming-based algorithm, which is from an essentially different perspective from ours.

It is also worth noting that our offline problem under the FN setting is essentially the classical maximum k-coverage problem (or the BMC problem with unit-cost sets only and an integer budget k) where previously-accepted sets cannot be dropped. *Below shows a reduction to the BMC problem.*

- Partition the target interval of our problem into discrete small intervals by the boundary points of all the released sub-intervals.

- The small intervals are equivalent to the elements of a universal set U in which each element has a weight equal to the length of its corresponding small interval. The released sub-intervals are equivalent to the sets in the collection $\mathsf{S} = \{S_1, \cdots, S_n\}$. The objective remains the same.

1.1 Related Works

There are mainly two threads of related works. The first thread is about the Online Budgeted Maximum Coverage (OBMC) problem. In the setting with unit-cost sets only, Saha and Getoor (2009) gave a 4-competitive deterministic algorithm. Rawitz and Rosén (2021) presented a lower bound of $\Omega(\frac{1}{\sqrt{1-r}})$ and a $\frac{4}{1-r}$-competitive deterministic algorithm for OBMC. Li et al (2021) proposed a heuristic algorithm to solve the problem by taking advantage of reinforcement learning and local search. Buchbinder et al (2019) studied the online submodular maximization with preemption, which aims to maximize a submodular function over the set of elements. They gave a constant-competitive online algorithm for the special case where an algorithm can hold at most k elements. When each element of the online maximum k-coverage problem belongs to exactly two sets and the intersection of any two sets has a size at most one, we have the maximum k-vertex coverage problem and Ausiello et al (2012) showed a 2-competitive deterministic algorithm together with a lower bound $\frac{3}{2}$. The well-known Knapsack problem can be regarded as a special case of BMC where sets are pairwise disjoint. In the online Knapsack problem, a decision-maker with a fixed budget selects a subset of an online sequence of options (each with a weight and a value) so as to maximize the total value of those items selected. Vaze (2018) studied the online knapsack problem with expected capacity constraint and presented a $\frac{1}{4e}$-competitive and a $(1 - \frac{1}{e})$-competitive algorithms for the general and the special case where each item has a unit weight, respectively. Babaioff et al (2007) studied the online knapsack problem under the assumption that elements arrive in random order, and they provided a $(10e)$-competitive algorithm for the case of general weights and values and a e-competitive algorithm for the special case when all weights are equal. Moreover, the Removable Online Knapsack (ROK) problem is the OBMC with pairwise disjoint sets. Iwama and Taketomi (2002) presented a polynomial-time online algorithm for the special case 1-bin ROK with a competitive ratio of less than $\frac{1+\sqrt{5}}{2} \approx 1.618$, and they also presented a matching lower bound for that special case.

The *second thread* is about the online k-secretary problem, which was introduced by Kleinberg (2005) and aimed to select k out of n independent values for maximizing the expected sum of individual secretary values. Bateni et al (2013) studied a generalized version called the submodular secretary problem, which aims to maximize the expectation of a submodular function that defines the efficiency of selected candidates based on their overlapping skills. Our problem is similar to theirs as the objective function of our problem is also submodular (see the function $Len(\cdot)$ of our model later in Section 2). However, we focus on the adversarial release order of sub-intervals (i.e., secretaries) using

the worst-case analysis of deterministic algorithms, while Bateni et al (2013) focused on a random release order of secretaries using the average-case analysis of algorithms. Note that the online knapsack problem studied in Babaioff et al (2007) is a weighted version of the classical secretary problem, which is called the knapsack secretary problem. We refer interested readers to Chrobak et al (2007); Chin et al (2006); Alon et al (2009); Assadi et al (2019); Feldman and Zenklusen (2018); Feldman et al (2018); Albers and Ladewig (2021) for more related works.

Below summarizes our main results (note that our bound results are also summarized in Table 2, which is at the end of Section 5, for discussing our bound gaps.)

- We investigate the online maximum k-interval coverage problem under comprehensive settings. After formulating the problem mathematically, we present two polynomial-time optimal solutions to the offline problem, in which a new dynamic programming-based solution is added from our conference version. Further, we show that a dynamic programming-based algorithm could accelerate with a feature we observe in an optimal solution.
- Lower bounds on the competitive ratio are proposed for the problem under different settings. Two algorithms are added from our conference version to present our well-designed release scheme of sub-intervals.
- Two threshold-based online deterministic algorithms are proposed to tackle the online problem, both of which run in $O(n)$-time. With our well-designed parameter-dependent thresholds, our algorithms achieve provable competitive ratios close to the lower bound.
- We show that with some observed constant thresholds, our algorithms still guarantee nice constant bound on competitive ratios despite some worst-case performance loss. We also show that an online deterministic algorithm that accepts sub-intervals with more non-increasing thresholds cannot induce better performance in the worst-case scenario than our proposed algorithms.

The remaining parts of this paper are organized as follows. Section 2 formally formulates our problem and those settings concerned in this paper. Before showing our lower bounds for the online problem in Section 4, we propose optimal solutions to the offline problem in Section 3 to shed light on our online algorithm design in Section 5. Section 6 concludes this work.

2 Problem Statement and Modelling

An online sequence $\mathbb{V} = \{V_1, V_2, \cdots\}$ of sub-intervals of a large target interval[1] $[0, a]$ are released in an adversarial order to the decision-maker, in which $V_i =$

[1]This target interval represents the expertise distribution of each online worker, which is in the simplest but fundamental way. For example, a worker's expertise distribution is close to the left boundary of the target interval if her expertise is in combinatorial optimization, while another with expertise distributed close to the right boundary implies a game theory background.

$[o_i, d_i] \subseteq [0, a]$ for each $V_i \in \mathbb{V}$. Upon the arrival of each $V_i \in \mathbb{V}$, the decision-maker must make a decision whether to accept or reject V_i immediately and irrevocably. Due to the quota limit, the decision-maker can accept no more than k (≥ 2)[2] sub-intervals. Any two different sub-intervals $V_i, V_j \in \mathbb{V}$ may intersect (i.e., $[o_i, d_j] \cap [o_j, d_j] \neq \varnothing$) considering that the expertise of online workers may overlap in reality. In addition, each $V_i = [o_i, d_i]$ could appear at any part of $[0, a]$ considering that an online worker could be an expert at any part of the expertise pool, which is unknown before the worker's arrival. In this paper, $|\cdot|$ and $\|\cdot\|$ denote the cardinality and the L^1 norm of sets of sub-intervals, respectively. When there is only one sub-interval (say V_i) inside $\|\cdot\|$, we abuse notation to use $\|V_i\| \triangleq d_i - o_i$ to denote the length of V_i. Table 1 summarizes the key notation in this paper. Now, we formally define the settings studied in this paper.

Table 1 Notation in this paper.

notation	Descriptions
$[0, a]$	The target interval.
k	The maximum number of sub-intervals to accept.
$V_i \triangleq [o_i, d_i]$	The ith released sub-interval.
$\mathbb{V}_i \triangleq \{V_1, V_2, \cdots, V_i\}$	The sequence of the first i released sub-intervals.
$\chi(\mathbb{V}_n, k)$	The optimal solution (OPT) to the offline problem, given both the set $\mathbb{V}$ of offline sub-intervals and the quota k beforehand.
$\Phi(\mathbb{V}_i)$	The subset of $\mathbb{V}_i$ that are accepted by our algorithm. Particularly, $\Phi(\mathbb{V}_n)$ (or $\Phi(\mathbb{V})$) denotes the set of sub-intervals that are finally accepted by our solution.
$OPT(\mathbb{V})$	The set of sub-intervals that are finally accepted by an optimal solution, which is sometimes simplified as OPT when there is no ambiguity in the context.
$Len(V_i) \triangleq d_i - o_i$	The length of a sub-interval V_i.
$Len(U) \triangleq \|\bigcup_{V_i \in U} V_i\|$	The overall covered length of $[0, a]$ by all the sub-intervals in a given set U. Particularly, $Len(\Phi(\mathbb{V}))$ and $Len(OPT(\mathbb{V}))$ indicate the overall covered length by our algorithm and an optimal solution, respectively.

With respect to the length $\|V_i\|$ of each $V_i \in \mathbb{V}$, we consider three settings.

- **Unit Length (UL)**: $d_i - o_i = 1$[3].
- **Flexible Length (FL)**: $(d_i - o_i)$ varies in a known range $[1, m]$, in which $m > 1$ as the $m = 1$ case degenerates to the UL setting.
- **Arbitrary Length (AL)**:$(d_i - o_i)$ varies arbitrarily in $[0, a]$.

In addition, we also consider a generalized version of the UL setting, which is the **Unit Sum (US)** setting[4]: each $V_i \in \mathbb{V}$ is no longer restricted to one sub-interval, but a batch of a finite number of disjoint sub-intervals of $[0, a]$ whose

[2]When $k = 1$, our problem degenerates to the classical secretary problem without expertise sub-interval overlap.

[3]The length of each sub-interval is normalized with regard to a.

[4]This models the scenario where an online worker masters different domains of expertise. And we keep the same unit-sum for all the sub-intervals to show similar strength of all the online workers.

sum of lengths is equal to 1. Accordingly, k batches of sub-intervals can be accepted in the US setting.

With respect to the total number $|\mathbb{V}|$ of released sub-intervals, we consider the following two settings respectively.

- **Fixed Number (FN)**: $|\mathbb{V}|$ is known in advance as a constant $n \in \mathbb{N}^*$. We further restrict $n \geq k+1$ as otherwise (when $n \leq k$) an optimal solution (OPT) can be easily achieved by just accepting all sub-intervals.
- **Variable Number (VN)**: $|\mathbb{V}|$ is not known to us and may vary due to the execution of an online algorithm.

When two settings are linked by a "-", we refer to the case that the two settings hold together, e.g., the UL-FN setting indicates the setting where all sub-intervals have unit length and the total number of released sub-intervals is known in advance. Whenever we specify a single setting in one dimension, we do not distinguish among settings in the other dimension, e.g., the FN setting refers to the context as any setting in {UL-FN, FL-FN, AL-FN}.

Given a sequence $\mathbb{V} = \{V_1, V_2, \cdots\}$ of online sub-intervals of $[0, a]$, our problem can be formulated as follows.

$$\begin{aligned} \max_{U \subseteq \mathbb{V}} \quad & \| \bigcup_{V_i \in U} V_i \| \\ \text{s.t.} \quad & |U| \leq k \end{aligned}$$

in which our *objective* is to accept a subset $U \subseteq \mathbb{V}$ of no more than k sub-intervals (i.e., $|U| \leq k$) such that the cumulative length, which is denoted as $Len(U) \triangleq \| \bigcup_{V_i \in U} V_i \|$, of the parts of $[0, a]$ that are covered by accepted sub-intervals in U is maximized.

Given an instance $\mathbb{V}$ of online sequence of sub-intervals, we denote $ALG(\mathbb{V})$ and $OPT(\mathbb{V})$ as the set of sub-intervals that are accepted by an online deterministic algorithm ALG and an optimal offline solution OPT with complete information of all sub-intervals known beforehand, respectively. For $\rho \geq 1$, ALG is called ρ-competitive for the problem if the following Inequality (1) holds for every instance $\mathbb{V}$.

$$Len(OPT(\mathbb{V})) \leq \rho \cdot Len(ALG(\mathbb{V})). \tag{1}$$

in which $Len(OPT(\mathbb{V}))$ and $Len(ALG(\mathbb{V}))$ denote the overall covered length by OPT and ALG, respectively. We also say the competitive ratio of ALG is ρ for the problem. Further, when a number $\gamma \geq 1$ ensures that $\gamma \leq \rho$ holds for all deterministic online algorithms, we say γ is a lower bound of the competitive ratio for the problem.

3 Optimal Offline Solution

In the offline setting of the problem, the complete information of the sequence $\mathbb{V}$ of sub-intervals is given at the very beginning. Without loss of generality,

we suppose there are totally n sub-intervals released as $\mathbb{V}_n = \{V_1, V_2, ..., V_n\}$. In the following, we present offline solutions to the US and the AL (including UL and FL) settings, respectively. [5]

3.1 For UL, FL and AL Settings

Now, we introduce an $O(kn + n \log n)$-time dynamic programming approach that outputs an optimal solution (OPT) to the offline problem under UL, FL and AL settings. We present our solution, which consists of two phases, in the AL setting. One can easily adapt our offline solution to the UL and FL settings since both UL and FL are special cases of AL.

When making a decision on $V_i \in \mathbb{V}_n$, we observe two critical sub-intervals $V_{\psi(i)}$ and $V_{\phi(i)}$ as defined below.

Definition 1 $V_{\psi(i)} = \underset{\{V_j \in \mathbb{V}_{i-1} | o_j < o_i \leq d_j\}}{\arg\max} \{o_i - o_j\}$ *indicates the sub-interval in* $\mathbb{V}_{i-1}$ *that intersects with* V_i *and has the left-most start point.*

Definition 2 $V_{\phi(i)} = \underset{\{V_j \in \mathbb{V}_{i-1} | d_j < o_i\}}{\arg\min} \{o_i - d_j\}$ *indicates the sub-interval in* $\mathbb{V}_{i-1}$ *that is disjoint from but is closest to* V_i*.*

In UL, FL and AL settings, each element $V_i \in \mathbb{V}_n$ is a continuous sub-interval of the target interval $[0, a]$. The following Proposition 1 characterizes the options that OPT could follow after V_i, which helps figure out the covered length that some accepted sub-interval contributes to the solution.

Proposition 1 *As long as an offline OPT accepts* V_i *under the AL (or UL or FL) setting and quota allows, the next sub-interval pending to be accepted, is either* $V_{\psi(i)}$ *or a sub-interval in* $\{V_1, V_2, \cdots, V_{\phi(i)}\}$*.*

Proof When V_i is accepted by offline OPT, the next sub-interval to accept (denoted as $V_{\leftarrow,i}$) lies in set $\mathbb{V}_{i-1} = \{V_1, \cdots, V_{i-1}\}$ as OPT is supposed to accept sub-intervals in decreasing order of their subscripts. Obviously, $V_{\leftarrow,i}$ either intersects or does not intersect with V_i. When $V_{\leftarrow,i}$ intersects with V_i, we know $V_{\leftarrow,i}$ should be the sub-interval in $\mathbb{V}_{i-1}$ that contributes the most additional length to V_i, which is $V_{\psi(i)}$ in Proposition 1, as accepting any sub-interval in $\{V_{\psi(i)}, \cdots, V_{i-1}\}$ cannot increase the overall length of OPT; As the sub-interval $V_\phi(i)$ in Proposition 1 indicates the sub-interval in $\mathbb{V}_{i-1}$ that is the nearest one to V_i and does not intersect with V_i, we know $V_{\leftarrow,i}$ lies in $\{V_1, V_2, \cdots, V_{\phi(i)}\}$ when $V_{\leftarrow,i}$ does not intersect with V_i. □

Phase 1. Sort sub-intervals in $\mathbb{V}_n = \{V_1, V_2, \cdots, V_n\}$ in non-decreasing order of their end points, which can be done in $O(n \log n)$ time. We abuse

[5] We do not distinguish our offline solution between FN and VN settings since our solution performs optimally in each of them.

notation in the offline solution to denote $(V_1, V_2, \cdots, V_n)$ as the sequence of sorted sub-intervals, i.e., $d_1 \leq d_2 \leq \cdots \leq d_n$, and further $\mathbb{V}_i = \{V_1, \cdots, V_i\}$ as the first i sub-intervals in the sequence. Suppose, in Phase 2, the decision-maker accepts sub-intervals in $\mathbb{V}_n$ in decreasing order of the subscripts.

Phase 2. Denote $\chi(\mathbb{V}_n, k)$ as the overall covered length of the target interval by OPT. To make a decision on a sub-interval $V_i \in \mathbb{V}_i$ with quota j, OPT follows *Bellman equation* in (5) and (6). Initially, we have $i = n$ and $j = k$ satisfying $1 \leq k < n$. As follows, we discuss three cases.

- **Case 1.** $1 \leq j < i$[6]. We have two sub-cases.
  - **Case 1.1.** OPT rejects V_i. Then, OPT continues to check the remaining sub-intervals in $\mathbb{V}_{i-1}$, i.e.,

$$\chi(\mathbb{V}_i, j) = \chi(\mathbb{V}_{i-1}, j) \tag{2}$$

  - **Case 1.2.** OPT accepts V_i. Since $i \geq 2$, there exists at least one sub-interval pending to be checked after V_i is accepted, which further implies that either $V_{\phi(i)}$ or $V_{\psi(i)}$ or both exists. We further discuss three sub-cases.
    - **Case 1.2.1** $V_{\phi(i)}$ exists but not $V_{\psi(i)}$. By Proposition 1, OPT could further accept some sub-interval in $\{V_1, V_2, \cdots, V_{\phi(i)}\}$ where we note that all sub-intervals are disjoint with V_i. Hence, V_i will contribute its whole length to the output solution, i.e.,

$$\chi(\mathbb{V}_i, j) = d_i - o_i + \chi(\mathbb{V}_{\phi(i)}, j-1) \tag{3}$$

    - **Case 1.2.2.** $V_{\psi(i)}$ exists but not $V_{\phi(i)}$. By Proposition 1, OPT further accepts $V_{\psi(i)}$. Note that V_i may intersect with $V_{\psi(i)}$, where the intersection length is indicated by $\|V_i \cap V_{\psi(i)}\|$. To this end, we introduce a new equation $\kappa(\mathbb{V}_i, j)$ in Equation (6), which is an embedded Bellman equation of $\chi(\cdot, \cdot)$ (see Equation (5)). $\kappa(\mathbb{V}_i, j)$ aims to accept at most j sub-intervals from the given set $\mathbb{V}_i$ such that the overall covered length of the target interval $[0, a]$ is maximized, in which the last sub-interval $V_i \in \mathbb{V}_i$ is always accepted.[7] When computing $\chi(\mathbb{V}_i, j)$, we consider $\|V_i \cap V_{\psi(i)}\|$ as a contribution of $V_{\psi(i)}$ instead of V_i. Accordingly, we only count the part of V_i, which is indicated by $Len(V_i) - \|V_i \cap V_{\psi(i)}\|$, that does not overlap with $V_{\psi(i)}$ as V_i's contribution to our solution. [8] Hence, $\chi(\mathbb{V}_i, j)$ can be got as follows.

$$\chi(\mathbb{V}_i, j) = d_i - o_i - \|V_i \cap V_{\psi(i)}\| + \kappa(\mathbb{V}_{\psi(i)}, j-1). \tag{4}$$

[6] This implies that OPT remains quota j, which is not sufficient to accept all the remaining sub-intervals in $\mathbb{V}_i$. In addition, we have $i \geq 2$ in this case.

[7] Note that the major difference between $\kappa(\mathbb{V}_i, j)$ and $\chi(\mathbb{V}_i, j)$ is that $\kappa(\mathbb{V}_i, j)$ always accepts the last sub-interval V_i in $\mathbb{V}_i$ while $\chi(\mathbb{V}_i, j)$ does not necessarily.

[8] Our approach to counting the contribution of each accepted sub-interval guarantees that the length contribution of sub-intervals accepted in the future will be independent of any accepted sub-interval, which provides convenience in formulating the $\chi(\mathbb{V}_i, j)$ in Case 2.

∗ **Case 1.2.3.** Both $V_{\psi(i)}$ and $V_{\phi(i)}$ exist. OPT would further accept the one in $\{V_{\psi(i)}, V_{\phi(i)}\}$ that leads to larger $\chi(\mathbb{V}_i, j)$, i.e., $\chi(\mathbb{V}_i, j)$ is updated by the larger right-hand side (RHS) between Equations (3) and (4).

Between Cases 1.1 and 1.2, OPT will follow the one that achieves larger $\chi(\mathbb{V}_i, j)$, so as to further achieve the maximum overall covered length.

- **Case 2.** $1 \leq i \leq j$. OPT remains sufficient quota to accept all the sub-intervals in $\mathbb{V}_i$. Then, OPT accepts all sub-intervals in $\mathbb{V}_i$. Due to our approach to counting the length contribution of an accepted sub-interval, we have $\chi(\mathbb{V}_i, j) = Len(\mathbb{V}_i)$.
- **Case 3.** $j = 0$ or $i = 0$.[9] OPT has run out of its quota. Then, OPT stops accepting any sub-interval, i.e., $\chi(\mathbb{V}_i, j) = 0$.

In summary, our Bellman Equation (5) with its embedded Equation (6) is presented as follows.

$$\chi(\mathbb{V}_i, j) = \begin{cases} \| \bigcup_{V_j \in \mathbb{V}_i} V_j \|, & 1 \leq i \leq j; \\ \max\{\text{RHS of (3)}, \text{ RHS of (4)}, \text{ RHS of (2)}\}, & 1 \leq j < i, V_{\psi(i)} \text{ and } V_{\phi(i)}; \\ \max\{\text{RHS of (3)}, \text{ RHS of (2)}\}, & 1 \leq j < i, \text{only } V_{\phi(i)}; \\ \max\{\text{RHS of (4)}, \text{ RHS of (2)}\}, & 1 \leq j < i, \text{only } V_{\psi(i)}; \\ 0, & j = 0 \text{ or } i = 0. \end{cases} \tag{5}$$

$$\kappa(\mathbb{V}_i, j) = \begin{cases} \| \bigcup_{V_j \in \mathbb{V}_i} V_j \|, & 1 \leq i \leq j; \\ \max\{\text{RHS of (3)}, \text{ RHS of (4)}\}, & 1 \leq j < i, V_{\psi(i)} \text{ and } V_{\phi(i)}; \\ \text{RHS of (3)}, & 1 \leq j < i, \text{only } V_{\phi(i)}; \\ \text{RHS of (4)}, & 1 \leq j < i, \text{only } V_{\psi(i)}; \\ 0, & j = 0 \text{ or } i = 0. \end{cases} \tag{6}$$

Remark 1 By scanning the sorted sub-intervals in $\mathbb{V}_k$, we can get values of $\{\| \bigcup_{V_j \in \mathbb{V}_1} V_j \|, \| \bigcup_{V_j \in \mathbb{V}_2} V_j \|, ..., \| \bigcup_{V_j \in \mathbb{V}_k} V_j \|\}$ in $O(k)$ time.[10] Since our solution generates $O(k(n-k))$ intermediate states, in which each state runs in $O(1)$ time.[11] With the preliminary sorting step, our solution runs in $O(k(n-k) + n \log n)$ time in total.

[9] Note that $i = 0$ would never happen, which is ensured by Case 2 and the basic assumption $n \geq k + 1$. We include the condition $i = 0$ here for the sake of completeness.

[10] Note that Klee's algorithm Klee (1977) can compute the length of an union of x sub-intervals in $O(x \log x)$ time, which is based on sorting the intervals. Accordingly, one can compute the length of x sorted sub-intervals in $O(x)$ time by scanning the x sub-intervals.

[11] Since an iteration will stop when $|\mathbb{V}_i| \leq j$ (see Case 2 of our offline solution). The overall states of our solution is improved to $O(k(n-k))$.

3.2 For US Setting

In the US setting, each element $V_p \in \mathbb{V}_n$ consists of a batch of small sub-intervals. Without loss of generality, we suppose each $V_p \in \mathbb{V}_n$ contains exactly u small sub-intervals whose sum length equals one, i.e., $V_p = \{V_{p1}, ...V_{pu}\}$ with $\| \cup_{q=1}^{u} V_{pq}\| = 1$.[12] Now, we discuss how to adapt a dynamic programming to the US setting, together with analysis of its time complexity.

Phase 1. Though each $V_p \in \mathbb{V}_n$ may be discrete, note that each $V_{pq} \in V_p$ is a continuous sub-interval. First, we sort small sub-intervals of each $V_i \in \mathbb{V}_n$ in non-decreasing order of their end points, which can be done in $O(nu \log u)$ time since there are totally n elements in $\mathbb{V}_n$. Then, we sort elements in $\mathbb{V}_n$ in non-decreasing order of their right-most endpoints. We still abuse notation to denote the sorted sequence as $(V_1, ..., V_n)$. Further in Phase 2, OPT still accepts elements in $\mathbb{V}_n$ following decreasing order of their subscripts. Totally, Phase 1 can be done in $O(n \log n + nu \log u)$ time.

Remark 2 In the US setting, any two elements $V_p, V_q \in \mathbb{V}_n$ may overlap arbitrarily, making Proposition 1 infeasible. As a consequence, the following two issues arise, which decelerates the offline solution significantly.

- The length that future elements (of $\mathbb{V}_n$) contribute to the solution depends on the accepting history of the solution, see the Case 2 in the following Phase 2.
- No quick way to compute the length that an accepted element $V_i \in \mathbb{V}_n$ contributes to the solution, see the Case 3.2 in the following Phase 2.

Phase 2. Further, we denote Φ_x as the set of the first x elements that are accepted by OPT, particularly, $\Phi_0 = \varnothing$. Similar as our solution to the AL setting, we discuss three cases when making a decision on $\mathbb{V}_i$ with quota j (given that OPT has accepted $k - j$ elements that formulates Φ_{k-j}), which is as follows.

- **Case 1.** $j = 0$ or $i = 0$. OPT stops accepting new element in $\mathbb{V}_i$, i.e., $\chi(\mathbb{V}_i, j) = 0$.
- **Case 2.** $1 \leq i \leq j$. OPT remains sufficient quota to accept all the remaining elements in $\mathbb{V}_i$, i.e., $\Phi_k = \mathbb{V}_i \cup \Phi_{k-j}$ and

$$\chi(\mathbb{V}_i, j) = \| \bigcup_{V_p \in \mathbb{V}_i \cup \Phi_{k-j}} V_p \|. \tag{7}$$

- **Case 3.** $1 \leq j < i$. We have two sub-cases.
  - **Case 3.1.** OPT rejects V_i and continues to check the other elements in $\mathbb{V}_{i-1}$, i.e.,

$$\chi(\mathbb{V}_i, j) = \chi(\mathbb{V}_{i-1}, j) \tag{8}$$

[12] When some V_q contains less than u of small sub-intervals, one can copy a small sub-interval of V_q several times until the total number of small sub-intervals of V_q reaches u, which does not affect the length contribution of V_q to the solution.

– **Case 3.2.** OPT accepts V_i and continues to check the other elements in $\mathbb{V}_{i-1}$, i.e., $\Phi_{k-j+1} = \Phi_{k-j} \cup \{V_i\}$ and

$$\chi(\mathbb{V}_i, j) = \chi(\mathbb{V}_{i-1}, j-1) + \underbrace{\| \bigcup_{V_p \in \Phi_{k-j} \cup \{V_i\}} V_p \| - \| \bigcup_{V_q \in \Phi_{k-j}} V_q \|}_{\text{the length that } V_i \text{ contributes}} \quad (9)$$

Thus, the Bellman Equation in the dynamic programming approach for the US setting is summarized as follows.

$$\chi(\mathbb{V}_i, j) = \begin{cases} \| \bigcup_{V_p \in \mathbb{V}_i \cup \Phi_{k-j}} V_p \|, & 1 \le i \le j; \\ \max\{\text{RHS of (8)}, \text{RHS of (9)}\}, & 1 \le j < i, \\ 0, & j = 0 \text{ or } i = 0. \end{cases} \quad (10)$$

Remark 3 Since the $\| \bigcup_{V_p \in \mathbb{V}_i \cup \Phi_{k-j}} V_p \|$ of Case 2 (of the US setting) depends on the accepting history Φ_{k-j}, we need to apply Klee's algorithm Klee (1977) to compute the length of the union of sub-intervals. Totally, our offline solution for the US setting generates $O(k(n-k))$ states, in which a state can be computed in $O(ku \log(ku))$ time by Klee's algorithm. Together with Phase 1, our solution to the US setting runs in $O(k^2(n-k)u \log(ku) + n \log n + nu \log u)$ time, where u denotes the number of small sub-intervals in an element of $\mathbb{V}_n$. When $u = 1$, we have a running time of $O(n \log n + k^2(n-k) \log k)$ of this solution to the AL setting, which implies that Proposition 1 helps accelerate dynamic programming-based solution.

4 Lower Bounds

In this section, we construct lower bounds on the competitive ratio for the settings studied in this paper, respectively.

Theorem 1 *In the AL setting, no online deterministic algorithm can achieve a bounded competitive ratio.*

Proof Our proof is based on a well-designed release scheme of sub-intervals (see Algorithm 4), under which we prove that the competitive ratio of any online deterministic algorithm could be arbitrarily large. Intuitively, by leveraging the release scheme, the adversary can always create opportunities for the OPT to benefit from ALG's loss. Other than by rejecting a sub-interval, ALG may lose some length by missing some sub-interval due to its quota limit. Now, we construct n sub-intervals as follows

$$V_i \triangleq \begin{cases} [0, \varepsilon^{k+1-i}], & \text{for } i = 1, 2, \cdots, k \\ [0, 1], & \text{for } i = k+1, ..., n \end{cases} \quad (11)$$

The following Algorithm 4 gives the release scheme of sub-intervals. To distinguish those sub-intervals in the scheme, we refer to those (in Equation (11)) that are

tentatively released as *original sub-interval* and those that have been updated (in Step 5 of Algorithm 4) as *updated sub-intervals*. We further discuss two cases.

Algorithm 1 Release Scheme of AL Sub-intervals

```
Tentatively release the n sub-intervals in (11).
for V_j ∈ {V_1, ..., V_k} do
    if ALG rejects V_j then
        for V_i ∈ {V_{j+1}, ..., V_n} do
            V_i ← [0, ε^{k+1−(j−1)}].
        end for
        break
    end if
end for
```

Case 1. ALG rejects some original sub-interval $V_j = [0, \varepsilon^{k+1-j}] \in \mathbb{V}_k$.

Due to the release scheme in Algorithm 4, the future sub-intervals are released as $[0, \varepsilon^{k+1-(j-1)}]$. Hence, ALG could achieve an overall length of at most $\varepsilon^{k+1-j+1}$ from accepted original sub-intervals $\{V_1, ..., V_{j-1}\}$ and updated sub-intervals. In contrast, OPT could achieve an overall length of at least ε^{k+1-j}, which is by accepting the original sub-interval V_j. When $\varepsilon \to 0$, we have the ratio follows

$$\rho \leq \frac{Len(V_j)}{Len(V_{j-1})} = \frac{\varepsilon^{k+1-j}}{\varepsilon^{k+1-j+1}} = \frac{1}{\varepsilon} \to +\infty$$

Case 2. ALG accepts all the k original sub-intervals.

Due to the release scheme in Algorithm 4, the future sub-intervals are released as $[0, 1]$. Then, OPT could achieve an overall length of 1 by accepting any updated sub-interval $[0, 1]$. In contrast, ALG achieves an overall length exactly equal to ε by accepting those k original sub-intervals. When $\varepsilon \to 0$, we have the ratio follows

$$\rho \leq \frac{Len(V_n)}{Len(V_k)} = \frac{1}{\varepsilon} \to +\infty$$

The proof completes. □

Theorem 2 *In the UL-FN setting, no online deterministic algorithm can achieve a competitive ratio better than (12)*

$$\begin{cases} \sqrt{2}, & \text{if } k = 2 \\ \min\{k^{\frac{1}{k}}, \frac{k^{\frac{\alpha}{k}}+k-\alpha-1}{k^{\frac{\alpha}{k}}+k-\alpha-2}, \frac{k}{k^{\frac{\alpha}{k}}+k-\alpha-1}\}, & \text{if } 3 \leq k \leq n-\alpha-1 \\ \min\{k^{\frac{1}{k}}, \frac{k^{\frac{\alpha}{k}}+k-\alpha-1}{k^{\frac{\alpha}{k}}+k-\alpha-2}, \frac{n-\alpha+2+k^{\frac{\alpha}{k}}-k^{\frac{n-k}{k}}}{k^{\frac{\alpha}{k}}+k-\alpha-1}\}, & \text{if } n-\alpha \leq k \leq n-1 \end{cases} \tag{12}$$

in which $\alpha = \left\lfloor 1 - \frac{\log(k^{\frac{1}{k}}-1)}{\log(k^{\frac{1}{k}})} \right\rfloor$

Proof Given the number k of quota (i.e., the maximum number of sub-intervals to accept), the number n ($\geq k+1$) of released sub-intervals of the target interval $[0, a]$ with the right endpoint a chosen as a large number, we discuss two cases with regard to the value of the quota k.

Case 1. ($k = 2$). We prove the lower bound in a constructed way as follows. Suppose $V_1 = [0, 1]$ and $V_2 = [\sqrt{2} - 1, \sqrt{2}]$, we have three sub-cases.

Case 1.1. ALG accepts both V_1 and V_2.

Further, the following $(n-2)$ sub-intervals are released.

$$\{V_j = [1, 2] \mid V_j \in \{V_3, ..., V_n\}\}$$

Further, the competitive ratio follows $\rho = \frac{2}{\sqrt{2}} = \sqrt{2}$ as OPT could accept V_1 and V_3.

Case 1.2. ALG accepts V_2 and rejects V_1.

Further, the following $(n-2)$ sub-intervals are released instead.

$$\{V_j = [\sqrt{2} - 1, \sqrt{2}] \mid V_j \in \{V_3, ..., V_n\}\}$$

Thus, we get $\rho = \frac{\sqrt{2}}{1}$ because OPT could accept V_1 and V_2;

Case 1.3. ALG rejects V_2.

Then, the future $(n-2)$ sub-intervals arrive as

$$\{V_j = [0, 1] \mid V_j \in \{V_3, ..., V_n\}\}$$

Hence, $\rho = \frac{\sqrt{2}}{1}$ as OPT could accept V_1 and V_2.

Case 2. ($3 \leq k \leq n-1$). Before presenting the lower bound of this case, we construct n sub-intervals as follows,

$$\{V_j \triangleq [\sum_{i=0}^{j-1} \xi_i, \sum_{i=0}^{j-1} \xi_i + 1] \Big| j \in \{1, ..., n\}\} \tag{13}$$

in which

$$\xi_i = \begin{cases} 0, & i = 0 \\ k^{\frac{i}{k}} - k^{\frac{i-1}{k}}, & 1 \leq i \leq \alpha \\ 1, & \alpha + 1 \leq i \leq n-1 \end{cases} \tag{14}$$

in which $\alpha = \left\lfloor 1 - \frac{k \log(k^{\frac{1}{k}} - 1)}{\log k} \right\rfloor$ with $3 \leq \alpha \leq k$ for $3 \leq k$, and $r_1 < r_2 < \cdots < r_k$. We let a be larger than $1 + \sum_{i=0}^{n-1} \xi_i$, which guarantees the above $V_1, \cdots, V_n$ are all sub-intervals of the target interval $[0, a]$. For each $i \in \{\alpha + 1, ..., n-1\}$, Equation (14) implies

$$\sum_{j=0}^{i} \xi_j = 1 + \sum_{j=0}^{i-1} \xi_j \tag{15}$$

in other words, the end point d_i ($= 1 + \sum_{j=0}^{i-1} \xi_j$) of each sub-interval $V_i \in \{V_{\alpha+1}, ..., V_{n-1}\}$ is exactly the start point o_{i+1} ($= \sum_{j=0}^{i} \xi_j$) of the following sub-interval V_{i+1}.

Now, we are ready to prove the lower bound of Case 2, which is due to the following release scheme as given in Algorithm 2. Intuitively, the release scheme tentatively releases the n sub-intervals as designed in (13). However, the release scheme changes as soon as ALG rejects some sub-interval of $\{V_1, ..., V_k\}$. If ALG rejects V_1 upon V_1's release, each of the future sub-intervals $\{V_2, V_3, ..., V_n\}$ is updated by $[1, 2]$. If ALG rejects some $V_j \in \{V_2, ..., V_n\}$ upon the release of V_j, each of future sub-intervals $\{V_{j+1}, ..., V_n\}$ is updated by $[\sum_{i=0}^{j-2} \xi_i, \sum_{i=0}^{j-2} \xi_i + 1]$ which is V_{j-1}. To distinguish

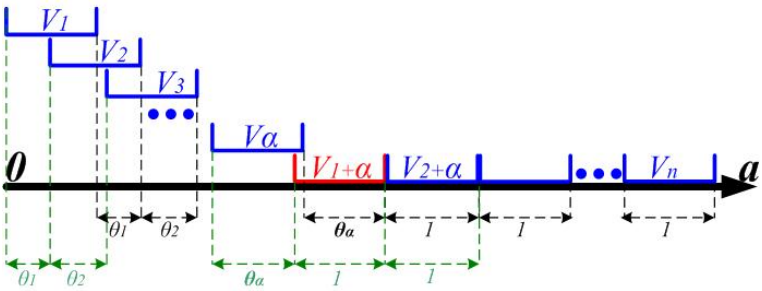

Fig. 1 Configuration of the sub-intervals in the lower bound of the UL-FN setting.

those sub-intervals in our release scheme, we refer to those (in Equation (13)) that are tentatively released as *original sub-intervals* and those that have been updated (in Steps 4 and 10 of Algorithm 2) in our release scheme as *updated sub-intervals*. We further discuss three cases.

Algorithm 2 Release Scheme of UL-FN Sub-intervals

1: Tentatively release the n sub-intervals $\{V_1, V_2, V_3, ..., V_n\}$ in (13) in increasing order of their subscripts.
2: **if** ALG rejects V_1 **then**
3: **for** $V_i \in \{V_2, ...V_n\}$ **do**
4: $V_i \leftarrow [1, 2]$.
5: **end for**
6: **else**
7: **for** $V_j \in \{V_2, ...V_k\}$ **do**
8: **if** ALG rejects V_j **then**
9: **for** $V_i \in \{V_{j+1}, ..., V_n\}$ **do**
10: $V_i \leftarrow [\sum_{i=0}^{j-2} \xi_i, 1 + \sum_{i=0}^{j-2} \xi_i]$.
11: **end for**
12: **end if**
13: **break**
14: **end for**
15: **end if**

Case 2.1. ALG rejects the original sub-interval V_1.

Due to the release scheme in Algorithm 2, ALG can only accept a sub-interval $[1, 2]$. As a result, ALG could achieve an overall covered length of at most 1, which implies $\rho = 2$ as OPT can get an overall length of 2 by accepting both V_1 and another $k - 1$ sub-intervals in $\{V'_2, ..., V'_n\}$.

Case 2.2. ALG accepts original sub-intervals $\{V_1, V_2, ..., V_{j-1}\}$ and rejects some original sub-interval $V_j \in \{V_2, ..., V_k\}$).

By our release scheme, each updated sub-interval $V_i \in \{V_{j+1}, ..., V_n\}$ has the same configuration as the original sub-interval V_{j-1}, which is $[\sum_{x=0}^{j-2} \xi_x, 1+\sum_{x=0}^{j-2} \xi_x]$. Accordingly, ALG can get an overall length of $1 + \sum_{i=0}^{j-2} \xi_i$ by original sub-intervals

$\{V_1, ..., V_{j-1}\}$. By accepting all the first j original sub-intervals in $\{V_1, ..., V_j\}$, OPT could achieve an overall length of $1 + \sum_{i=0}^{j-1} \xi_i$. Thus, the ratio follows

$$\rho = \frac{1 + \xi_1 + \cdots + \xi_{j-2} + \xi_{j-1}}{1 + \xi_1 + \cdots + \xi_{j-2}}. \tag{16}$$

Denote $\alpha = \left\lfloor 1 - \frac{k \log(k^{\frac{1}{k}} - 1)}{\log k} \right\rfloor$, we further discuss the ratio in two sub-cases.

Case 2.2.1. ($j \leq \alpha$). By Equations (14) and (16), we get

$$\begin{aligned}\rho &= \frac{1 + (k^{\frac{1}{k}} - 1) + \cdots + (k^{\frac{j-1}{k}} - k^{\frac{j-2}{k}}) + (k^{\frac{j}{k}} - k^{\frac{j-1}{k}})}{1 + (k^{\frac{1}{k}} - 1) + \cdots + (k^{\frac{j-1}{k}} - k^{\frac{j-2}{k}})} \\ &= \frac{k^{\frac{j}{k}}}{k^{\frac{j-1}{k}}} = k^{\frac{1}{k}} < 1.5 < 2\end{aligned}$$

Case 2.2.2. ($\alpha + 1 \leq j \leq k$). By Equations (14) and (16), we have

$$\begin{aligned}\rho &= \frac{1 + (k^{\frac{1}{k}} - 1) + \cdots + (k^{\frac{\alpha}{k}} - k^{\frac{\alpha-1}{k}}) + j - \alpha - 1}{1 + (k^{\frac{1}{k}} - 1) + \cdots + (k^{\frac{\alpha}{k}} - k^{\frac{\alpha-1}{k}}) + j - \alpha - 2} \\ &= \frac{k^{\frac{\alpha}{k}} + j - \alpha - 1}{k^{\frac{\alpha}{k}} + j - \alpha - 2} \\ &\geq \frac{k^{\frac{\alpha}{k}} + k - \alpha - 1}{k^{\frac{\alpha}{k}} + k - \alpha - 2}\end{aligned}$$

in which the inequality holds by the basic condition of this case.

Case 2.3. ALG accepts the first k original sub-intervals $\{V_1, V_2, V_3, ..., V_k\}$.

Thus, ALG could achieve an overall length of

$$Len(\Phi(\mathbb{V})) = k^{\frac{\alpha}{k}} + k - \alpha - 1 \tag{17}$$

Later, the original sub-intervals $\{V_{k+1}, ..., V_n\}$ are pending to be released. We have two cases.

Case 2.3.1. ($\alpha+k+1 \leq n$). By Equation (15) and $n-k+1 \geq \alpha+2$, OPT could achieve an overall length of k by accepting those original sub-intervals in $\{V_{n-k+1}, ..., V_n\}$. With Equation (17), the ratio follows $\rho = \frac{k}{k^{\frac{\alpha}{k}} + k - \alpha - 1}$.

Case 2.3.2. ($1 + k \leq n \leq \alpha + k$). Equation (14) implies

$$\xi_0 < \cdots < \xi_{\alpha+1} = \cdots = \xi_n = 1 \tag{18}$$

Since OPT performs no worse than accepting the last k sub-intervals $\{V_{n-k+1}, ..., V_n\}$, OPT could achieve an overall length of no less than $Len(\bigcup_{i=n-k+1}^{n} V_i)$ which follows

$$\begin{aligned}&Len(\bigcup_{i=n-k+1}^{n} V_i) \\ &= Len(\bigcup_{i=\alpha+2}^{n} V_i) + Len(\bigcup_{i=n-k+1}^{\alpha+1} V_i) \\ &= Len(\bigcup_{i=\alpha+2}^{n} V_i) + Len(\bigcup_{i=1}^{\alpha+1} V_i) - \sum_{i=1}^{n-k} \xi_i\end{aligned}$$

$$= (n - \alpha + 1) + k^{\frac{\alpha}{k}} - \sum_{i=1}^{n-k} \xi_i$$
$$= n - \alpha + 2 + k^{\frac{\alpha}{k}} - k^{\frac{n-k}{k}}$$

in which the first equation holds by Equation (15) and the second equation holds by Equation (14) (see **Fig. 1**). This further implies the ratio

$$\rho \geq \frac{n - \alpha + 2 + k^{\frac{\alpha}{k}} - k^{\frac{n-k}{k}}}{k^{\frac{\alpha}{k}} + k - \alpha - 1}$$
$$= \frac{k^{\frac{\alpha}{k}} + k - \alpha - 1 + (n - k + 3 - k^{\frac{n-k}{k}})}{k^{\frac{\alpha}{k}} + k - \alpha - 1}$$

In summary, the competitive ratio of an online algorithm follows Equation (12) in which $\beta = \min\{k, n - \alpha + 2 + k^{\frac{\alpha}{k}} - k^{\frac{n-k}{k}}\}$ and $\alpha = \left\lfloor 1 - \frac{k \log(k^{\frac{1}{k}} - 1)}{\log k} \right\rfloor$. □

Corollary 1 *For UL-VN, no online deterministic algorithm can achieve a competitive ratio better than*

$$\begin{cases} \sqrt{2}, & \text{if } k = 2 \\ \min\{k^{\frac{1}{k}}, \frac{k^{\frac{\alpha}{k}} + k - \alpha - 1}{k^{\frac{\alpha}{k}} + k - \alpha - 2}, \frac{k}{k^{\frac{\alpha}{k}} + k - \alpha - 1}\}, & \text{if } 3 \leq k \end{cases} \tag{19}$$

in which $\alpha = \left\lfloor 1 - \frac{\log(k^{\frac{1}{k}} - 1)}{\log(k^{\frac{1}{k}})} \right\rfloor$.

Proof In the UL-VN setting, the overall number $|\mathbb{V}|$ of released sub-intervals is not given to the ALG beforehand, and ALG could learn a sub-interval only upon the sub-interval's release. In fact, the adversary could release an arbitrary number of sub-intervals to stop ALG achieving a good worst-case performance. By applying a similar release scheme of sub-intervals as in the Theorem 2, where the Case 2.3.2 of Theorem 2 is further removed since we can simply bound the overall length of OPT by k, and the other cases of Theorem 2 largely remain, the lower bound of UL-VN follows

$$\begin{cases} \sqrt{2}, & \text{if } k = 2 \\ \min\{k^{\frac{1}{k}}, \frac{k^{\frac{\alpha}{k}} + k - \alpha - 1}{k^{\frac{\alpha}{k}} + k - \alpha - 2}, \frac{k}{k^{\frac{\alpha}{k}} + k - \alpha - 1}\}, & \text{if } 3 \leq k \end{cases}$$

in which $\alpha = \left\lfloor 1 - \frac{k \log(k^{\frac{1}{k}} - 1)}{\log k} \right\rfloor$. □

Theorem 3 *For FL-FN, no online deterministic algorithm can achieve a competitive ratio better than* $\frac{2km}{2km - (m-1)\min\{k, n-k\}}$*, which is strictly smaller than 2.*

Proof In FL-FN, the length of each sub-interval $V_i = [o_i, d_i]$ varies in a given range $[1, m]$, i.e., $\|V_i\| \in [1, m]$. The lower bound is constructed as follows. The first $\tau \triangleq \min\{k, n - k\}$ sub-intervals are released as $\{V_i = [i - 1, i] \mid i \in \{1, 2, ..., \tau\}\}$. Suppose ALG accepts x (which varies from 0 to τ) out of the τ sub-intervals, we discuss two

cases.

Case 1. $(0 < x \leq \lfloor \frac{\tau}{2} \rfloor)$. Then, all the future sub-intervals are released as $[0, 1]$. As a result, ALG could get an overall length of at most $\lfloor \frac{\tau}{2} \rfloor$. Hence, we have $\rho \geq \frac{\tau}{\lfloor \frac{\tau}{2} \rfloor} \geq 2$;

Case 2. $(\lceil \frac{\tau}{2} \rceil \leq x \leq \tau)$. Then, the remaining $(n - \tau)$ sub-intervals arrive as

$$\{V_j = [\tau + (j - \tau - 1)m, \tau + (j - \tau)m] \mid j \in \{\tau + 1, \tau + 2, ..., n\}\}. \tag{20}$$

As a result, ALG could get an overall length of at most $x + (k - x) \cdot m$, which is by those x accepted sub-intervals and another $(k-x)$ sub-intervals from the last $(n-\tau)$ sub-intervals in (20). Since $n-\tau \geq k$, OPT could achieve an overall length of km by accepting k out of those $(n - \tau)$ sub-intervals in (20). Thus,

$$\rho \geq \frac{km}{km - x(m-1)} \geq \frac{2km}{2km - (m-1)\min\{k, n-k\}}.$$

Therefore, no online algorithm can achieve a competitive ratio better than

$$\min\{2, \frac{2km}{2km - (m-1)\min\{k, n-k\}}\} = \frac{2km}{2km - (m-1)\min\{k, n-k\}}$$

The proof completes. □

Corollary 2 *For FL-VN, no online deterministic algorithm can achieve a competitive ratio better than* $\frac{2m}{m+1}$ *which is strictly smaller than 2.*

Proof By setting $\tau = k$, removing $n = |\mathbb{V}|$ in the release scheme of sub-intervals in Theorem 3, and simply bounding the overall length of OPT by km (in which k is the quota constraint and m denotes the largest length of a released sub-interval in $\mathbb{V}$), one can easily derive a lower bound of $\frac{km}{km - \frac{k}{2}(m-1)} = \frac{2m}{m+1}$ for the FL-VN setting. □

Corollary 3 *For US-FN, no online deterministic algorithm can achieve a competitive ratio better than (12), where* $\alpha = \left\lfloor 1 - \frac{\log(k^{\frac{1}{k}} - 1)}{\log(k^{\frac{1}{k}})} \right\rfloor$.

Proof By partitioning each sub-interval $V_i \in \mathbb{V}$ of the unit-length case arbitrarily into a finite number of disjoint sub sub-intervals, we can get an instance of the unit-sum case. Hence, the lower bound, which is showed in Theorem 2, applies to the unit-sum case as well. □

5 Upper bounds

In this section, we present two online deterministic algorithms, in which the key insight is to seek a trade-off between the quota and the loss from missed sub-intervals.

5.1 Single-threshold Online Algorithm

We first investigate a Single-threshold Online Algorithm (SOA), in which the idea is to accept a sub-interval if the added length without overlap exceeds a predefined threshold. SOA only fits the FN setting, but with some slight changes, it can be extended to tackle the VN setting, as introduced later in this subsection. To guarantee good worst-case performance, one has to design the threshold of SOA subtly, which will be discussed later when we formally introduce SOA. With our well-designed threshold, SOA and SOA_{VN} can achieve competitive ratios strictly smaller than 2 for the FN and the VN settings, respectively.

Algorithm 3 Single-threshold Online Algorithm (SOA)

Input: A sequence $\mathbb{V} = \{V_1, V_2, ..., V_n\}$ of n sub-intervals of the target interval $[0, a]$, in which $V_i = [o_i, d_i]$ for each $V_i \in \mathbb{V}$, the quota k ($2 \leq k \leq n-1$).
Output: A set of accepted sub-intervals, i.e., $\Phi(\mathbb{V}_n)$.

$\Phi(\mathbb{V}_1) = \{V_1\}$; ▷ always accept V_1
for $i = 2;\ i++;\ i \leq n$ **do**
  if $|\Phi(\mathbb{V}_{i-1})| = k$ **then**
    $\Phi(\mathbb{V}_n) = \Phi(\mathbb{V}_{i-1})$;
    break; ▷ Stop accepting sub-intervals as SOA runs out of the quota
  else if $k - |\Phi(\mathbb{V}_{i-1})| \geq n - i + 1$ **then**
    $\Phi(\mathbb{V}_i) = \Phi(\mathbb{V}_{i-1}) \cup \{V_i\}$; ▷ accept V_i by the quota-enough
  else
    if $Len(\Phi(\mathbb{V}_{i-1}) \cup \{V_i\}) - Len(\Phi(\mathbb{V}_{i-1})) \geq \theta$ **then** ▷ θ refers to (36)
      $\Phi(\mathbb{V}_i) = \Phi(\mathbb{V}_{i-1}) \cup \{V_i\}$; ▷ accept V_i by threshold-accept
    else
      $\Phi(\mathbb{V}_i) = \Phi(\mathbb{V}_{i-1})$; ▷ reject V_i
    end if
  end if
end for

Now, we formally introduce SOA algorithm as follows. *First,* SOA always accepts the first released sub-interval V_1. *Then,* for each future sub-interval $V_i \in \{V_2, ..., V_n\}$, SOA accepts V_i only when it satisfies one of the following two conditions: (i) **Quota-enough condition**, after accepting V_i, SOA has enough quota to accept all the future sub-intervals, i.e., $k - |\Phi(\mathbb{V}_{i-1})| \geq n - i + 1$; (ii) **Threshold-accepting condition**, SOA remains quota (i.e., $k - |\Phi(\mathbb{V}_{i-1})| \geq 1$) and V_i contributes an additional length of at least θ to the current solution, i.e., $Len(\Phi(\mathbb{V}_{i-1}) \cup \{V_i\}) - Len(\Phi(\mathbb{V}_{i-1})) \geq \theta$. The pseudocode of the SOA is given in Algorithm 3. Before discussing the selection of θ, we observer the followings regarding SOA, which help derive Theorem 5 and further select the threshold θ.

Observation 1 SOA always accepts k sub-intervals, including V_1. And SOA can only accept V_n by the quota-enough condition.

Observation 2 When some sub-interval meets the quota-enough condition, all the future sub-intervals meet the quota-enough condition as well. When SOA finally rejects V_n, all the k accepted sub-intervals meet the threshold-accepting condition.

Intuitively, when the threshold θ is set small, SOA possibly uses up its quota very early and misses sub-intervals released in a late period. Conversely, when θ is set large, SOA possibly misses many sub-intervals (which could have contributed a lot to SOA) released in an early period. The following Theorem 4 captures how θ affects the worst-case performance of SOA, which provides a foundation for finding the best-fit θ later.[13]

Theorem 4 *For UL-FN, SOA runs in $O(n)$ time and achieves a competitive ratio no larger than*

$$\max\{1+2\theta, \min\{\frac{k}{1+(k-1)\theta}, 1+\frac{n-k}{1+(k-1)\theta}\}\}. \tag{21}$$

Proof SOA runs in $O(n)$ time as it runs in no more than n iterations in which each iteration runs in $O(1)$ time. When bounding the competitive ratio of SOA, we find whether SOA accepts V_n is critical. Accordingly, we discuss two cases.

Case 1. V_n is accepted.

This implies that SOA accepts some sub-intervals by the quota-enough condition, see Observations 1 and 2. Suppose $V_i \in \{V_2, V_3, ..., V_n\}$ is the first sub-interval that is accepted by the quota-enough condition, which implies that all sub-intervals in $\{V_i, V_{i+1}, ..., V_n\}$ are accepted by the quota-enough condition as well. However, these $n-i+1$ sub-intervals in $\{V_i, V_{i+1}, ..., V_n\}$ would only contribute zero length to SOA in the worst-case scenario. Accordingly, SOA only benefits from those sub-intervals $\Phi(\mathbb{V}_{i-1})$ that are accepted (by the threshold-accepting condition) before V_i, i.e.,

$$Len(\Phi(\mathbb{V}_n)) \geq Len(\Phi(\mathbb{V}_{i-1})) \tag{22}$$

Suppose $Len(\Phi(\mathbb{V}_{i-1}))$ consists of a number ζ of disjoint intervals, which are denoted by $\overline{V}_1$, $\overline{V}_2$,..., $\overline{V}_\zeta$ respectively. Clearly, $1 \leq x \leq k$. Accordingly, $Len(\overline{V}_1)$,..., $Len(\overline{V}_\zeta)$ denote the length of those disjoint intervals of $Len(\Phi(\mathbb{V}_{i-1}))$, respectively. As such,

$$Len(\Phi(\mathbb{V}_{i-1})) = \sum_{i=1}^{\zeta} Len(\overline{V}_i) \tag{23}$$

Due to the threshold-accepting condition, each sub-interval rejected by SOA contributes an additional length of less than θ to $Len(\Phi(\mathbb{V}_{i-1}))$, as otherwise it would have been accepted. SOA totally rejects $n-k$ sub-intervals that violate the threshold-accepting condition in this case, implying

$$Len(OPT) \leq Len(\Phi(\mathbb{V}_n)) + \theta(n-k) \tag{24}$$

[13] Here, the "best-fit θ" refers to the θ that minimizes the upper bound as presented in Theorem 4.

Those ζ disjoint intervals (formed by those accepted sub-intervals) can create at most 2ζ gaps, which can be exploited by those rejected sub-intervals. Thus, we have

$$Len(OPT) \leq Len(\Phi(\mathbb{V}_n)) + 2\zeta \cdot \theta \tag{25}$$

In summary, the overall covered length of OPT follows

$$Len(OPT) \leq Len(\Phi(\mathbb{V}_n)) + \min\{2\theta\zeta, \theta(n-k)\} \tag{26}$$

Further, the competitive ratio of SOA of this case follows

$$\begin{aligned} \rho &= \frac{Len(OPT)}{Len(\Phi(\mathbb{V}_n))} \\ &\leq 1 + \frac{\min\{2\theta\zeta, \theta(n-k)\}}{Len(\Phi(\mathbb{V}_n))} \\ &\leq 1 + \frac{2\theta\zeta}{\sum_{i=1}^{\zeta} Len(\overline{V}_i)} \\ &\leq 1 + 2\theta \end{aligned} \tag{27}$$

in which the first inequality holds by (26), the second inequality holds by Inequality (22) and Equation (23), and the last inequality holds because $Len(\overline{V}_i) \geq 1$ holds for each $\overline{V}_i \in \{\overline{V}_1, ..., \overline{V}_\zeta\}$.

Case 2. V_n is rejected.

This implies that all the k sub-intervals accepted by SOA meet the threshold-accepting condition, see Observation 2. Among those k sub-intervals accepted by SOA, V_1 contributes one to SOA while each of $\Phi(\mathbb{V}_n) - \{V_1\}$ contributes at least θ to SOA (see threshold-accepting condition). Thus, we have

$$Len(\Phi(\mathbb{V}_n)) \geq 1 + (k-1)\theta \tag{28}$$

Suppose V_i is the sub-interval that is finally accepted by SOA, i.e., $|\Phi(\mathbb{V}_i)| = k$ and $Len(\Phi(\mathbb{V}_n)) = Len(\Phi(\mathbb{V}_i))$, implying that SOA misses all sub-intervals in $\{V_{i+1}, ..., V_n\}$ which could be utilized by OPT. Since the algorithm totally rejects $n-k$ sub-intervals, OPT could aggregate a covered length that exceeds $Len(\Phi(\mathbb{V}_n))$ (what SOA achieves) at most $n-k$, i.e.,

$$Len(OPT) \leq Len(\Phi(\mathbb{V}_n)) + n - k \tag{29}$$

Due to the quota k, we have

$$Len(OPT) \leq k \tag{30}$$

In summary, the overall length accepted by OPT is bounded as follows

$$Len(OPT) \leq \min\{k, Len(\Phi(\mathbb{V}_n)) + n - k\} \tag{31}$$

Further, the competitive ratio of SOA follows.

$$\begin{aligned} \rho &= \frac{Len(OPT)}{Len(\Phi(\mathbb{V}_n))} \\ &\leq \min\{\frac{k}{Len(\Phi(\mathbb{V}_n))}, 1 + \frac{n-k}{Len(\Phi(\mathbb{V}_n))}\} \\ &\leq \min\{\frac{k}{1+(k-1)\theta}, 1 + \frac{n-k}{1+(k-1)\theta}\} \end{aligned} \tag{32}$$

in which the first and the second inequalities hold by Inequality (31) and Inequality (28), respectively.

By Inequalities (27) and (32), we get an upper bound on the competitive ratio of SOA as follows

$$\rho \leq \max\{1 + 2\theta, \min\{\frac{k}{1+(k-1)\theta}, 1 + \frac{n-k}{1+(k-1)\theta}\}\} \tag{33}$$

The proof completes. □

Threshold Selection in SOA with the UL-FN Setting. To find the best-fit threshold θ, we transform the upper bound (21) of Theorem 4 to the following.

$$\min\{\max\{1+2\theta, \frac{k}{1+(k-1)\theta}\}, 1+\max\{2\theta, \frac{n-k}{1+(k-1)\theta}\}\}\} \tag{34}$$

Since $1 \leq k < n$, we find the unique non-negative solution $\frac{\sqrt{9k^2-14k+9}-k-1}{4(k-1)}$ to equation $1+2\theta = \frac{k}{1+(k-1)\theta}$, and the unique non-negative solution $\frac{\sqrt{1+2(k-1)(n-k)}-1}{2k-2}$ to the equation $2\theta = \frac{n-k}{1+(k-1)\theta}$. Further, we find the threshold that minimizes the upper bound (21) of Theorem 4 as follows.

$$\theta = \min\{\frac{\sqrt{1+2(k-1)(n-k)}-1}{2k-2}, \frac{\sqrt{9k^2-14k+9}-k-1}{4(k-1)}\} \tag{35}$$

Based on Equation (35), the following proposition further characterizes the value of the threshold θ in SOA.

Proposition 2 *In SOA, the threshold is set as follows*

$$\theta = \begin{cases} \frac{\sqrt{1+2(k-1)(n-k)}-1}{2k-2}, & \text{for } \lceil\frac{667n}{1000}\rceil \leq k \leq n-1 \\ \frac{\sqrt{9k^2-14k+9}-k-1}{4(k-1)}, & \text{for } 2 \leq k \leq \lceil\frac{667n}{1000}\rceil - 1 \end{cases} \tag{36}$$

Proof First, we notice the following two inequalities by calculations,

$$\frac{\sqrt{1+2(\lceil\frac{667n}{1000}\rceil-1)(n-\lceil\frac{667n}{1000}\rceil)}-1}{2\lceil\frac{667n}{1000}\rceil-2} \leq \frac{\sqrt{9\lceil\frac{667n}{1000}\rceil^2-14\lceil\frac{667n}{1000}\rceil+9}-\lceil\frac{667n}{1000}\rceil-1}{4(\lceil\frac{667n}{1000}\rceil-1)}$$

and

$$\frac{\sqrt{1+2(\lceil\frac{667n}{1000}\rceil-2)(n-\lceil\frac{667n}{1000}\rceil+1)}-1}{2\lceil\frac{667n}{1000}-1\rceil-2}$$
$$\geq \frac{\sqrt{9(\lceil\frac{667n}{1000}\rceil-1)^2-14(\lceil\frac{667n}{1000}\rceil-1)+9}-\lceil\frac{667n}{1000}\rceil-2}{4(\lceil\frac{667n}{1000}-1\rceil-1)}$$

Further, we know $\frac{\sqrt{1+2(k-1)(n-k)}-1}{2k-2}$ decreases as k increases within $\{2,3,...,n-1\}$, while $\frac{\sqrt{9k^2-14k+9}-k-1}{4(k-1)}$ increases as k increases, given the number n of online sub-intervals. This completes the proof. □

The following theorem shows the upper bound of SOA algorithm with the best-fit threshold θ given by Equation (36)

Theorem 5 *With θ given by Equation (36), SOA achieves a competitive ratio no worse than* $\min\{\frac{\sqrt{1+2(k-1)(n-k)}-1}{k-1}+1, \frac{\sqrt{9k^2-14k+9}-k-1}{2(k-1)}+1\}$ *for the UL-FN setting.*

Proof By Theorem 4, we know SOA runs in $O(n)$ time. When V_n is accepted by SOA, Theorem 4 admits an upper bound of

$$1+2\theta = 1+\min\{\frac{\sqrt{1+2(k-1)(n-k)}-1}{k-1}, \frac{\sqrt{9k^2-14k+9}-k-1}{2(k-1)}\} \tag{37}$$

When V_n is rejected by SOA, we discuss two cases.

Case 1. ($\lceil\frac{667n}{1000}\rceil \le k \le n-1$). We have $\theta = \frac{\sqrt{1+2(k-1)(n-k)}-1}{2k-2}$ by Proposition 2. For each $k \in [\lceil\frac{667n}{1000}\rceil, n] \subseteq (\frac{8n+10-\sqrt{(8n-8)^2+132}}{16}, \frac{8n+10+\sqrt{(8n-8)^2+132}}{16})$, note that

$$\frac{\partial(\frac{\theta}{\frac{2k-n-1}{k-1}})}{\partial k} = \frac{8k^2-(8n+10)k+9n-3}{4(2k-n-1)^2\sqrt{1+2(k-1)(n-k)}} < 0$$

Further,

$$\frac{\theta}{\frac{2k-n-1}{k-1}} \le \frac{\theta}{\frac{2k-n-1}{k-1}}\Big|_{k=\lceil\frac{667n}{1000}\rceil} < 1 \tag{38}$$

By Theorem 4, the upper bound follows

$$\rho \le \min\{\frac{k}{1+(k-1)\theta}, 1+\frac{n-k}{1+(k-1)\theta}\}$$
$$= \frac{\sqrt{1+2(k-1)(n-k)}-1}{k-1}+1 \qquad \text{by (38) and } \theta = \frac{\sqrt{1+2(k-1)(n-k)}-1}{2k-2}$$

Case 2. ($2 \le k \le \lceil\frac{667n}{1000}\rceil - 1$). By Proposition 2,

$$\theta = \frac{\sqrt{9k^2-14k+9}-k-1}{4(k-1)} \le \frac{\sqrt{1+2(k-1)(n-k)}-1}{2k-2} \tag{39}$$

By Theorem 4 implies the upper bounds follows

$$\rho \le \min\{\frac{k}{1+(k-1)\theta}, 1+\frac{n-k}{1+(k-1)\theta}\}$$
$$\le \frac{k}{1+(k-1)\theta} = \frac{\sqrt{9k^2-14k+9}-k-1}{2(k-1)}+1 \quad \text{by } \theta = \frac{\sqrt{9k^2-14k+9}-k-1}{4(k-1)}$$

By Case 1 and Case 2, the proof completes. □

By applying a similar analysis as in Theorem 5, one can easily get the following Theorem 6 and Theorem 7 which give upper bounds of algorithm SOA in the US-FL and FL-FN settings, respectively.

Theorem 6 *With θ given by Equation (36), SOA runs in $O(n)$ time and achieves a competitive ratio no larger than* $\min\{\frac{\sqrt{1+2(k-1)(n-k)}-1}{k-1}+1, \frac{\sqrt{9k^2-14k+9}-k-1}{2(k-1)}+1\}$ *for the US-FN setting.*

Proof SOA runs in $O(n)$ time since it runs in no more than n iterations in which each runs in $O(1)$ time. In the US-FN setting, the sum of lengths of sub-intervals in each $V_i \in \mathbb{V}$ is 1. We discuss two cases.

Case 1. SOA accepts V_n.

Suppose $Len(\Phi(\mathbb{V}_n))$ consists of a number ζ of disjoint intervals. By Theorem 4, we can bound the overall covered length of OPT as follows

$$Len(OPT) \leq Len(\Phi(\mathbb{V}_n)) + \min\{2\theta\zeta, \theta(n-k)\} \tag{40}$$

which is due to the loss of SOA from rejected sub-intervals. Further, we have

$$\begin{aligned}\rho &\leq 1 + 2\theta \\ &\leq \min\{\frac{\sqrt{1+2(k-1)(n-k)}-1}{k-1} + 1, \frac{\sqrt{9k^2-14k+9}-k-1}{2(k-1)} + 1\}\end{aligned}$$

Case 2. SOA does not accept V_n.

On one hand, we can bound the overall covered length of SOA as follows

$$Len(\Phi(\mathbb{V}_n)) \geq 1 + (k-1)\theta \tag{41}$$

which is due to the threshold-accepting condition and Theorem 4. On the other hand, we can bound the overall covered length of OPT below.

$$Len(OPT) \leq \min\{k, Len(\Phi(\mathbb{V}_n)) + (n-k)\} \tag{42}$$

which is due to the quota k, the total number $n-k$ of sub-intervals that are either missed or rejected by SOA, and the sum length 1 of each $V \in \mathbb{V}$. Further, we can bound the competitive ratio of SOA of this case as follows

$$\begin{aligned}\rho &= \frac{Len(OPT)}{Len(\Phi(\mathbb{V}_n))} \\ &\leq \min\{\frac{\sqrt{1+2(k-1)(n-k)}-1}{k-1} + 1, \frac{\sqrt{9k^2-14k+9}-k-1}{2(k-1)} + 1\}\end{aligned}$$

By Cases 1-2, the proof completes. □

Theorem 7 *With the threshold given by Equation (48), SOA runs in $O(n)$ time and achieves a competitive ratio no larger than*

$$\min\{\frac{\sqrt{1+2(k-1)(n-k)m}-1}{k-1} + 1, \frac{\sqrt{(1+8m)k^2-(6+8m)k+9}-k-1}{2(k-1)} + 1\}$$

for the FL-FN setting, in which m indicates the maximum length of a sub-interval.

Proof The running time of SOA is proved to be $O(n)$ in Theorem 5. In the FL-FN setting, the length of each sub-interval varies in a fixed range $[1, m]$. We discuss two cases.

Case 1. SOA accepts V_n (the last sub-interval released).

Note that SOA rejects $n-k$ sub-intervals in this case, all of which violate the threshold-accepting condition of SOA. Accordingly, we can get the following bound by applying a similar proof idea of the Case 1 in Theorem 5.

$$\begin{aligned}\rho &\leq 1 + 2\theta \\ &\leq 1 + \min\{\frac{\sqrt{1+2(k-1)(n-k)m}-1}{k-1}, \frac{\sqrt{(1+8m)k^2-(6+8m)k+9}-k-1}{2(k-1)}\}\end{aligned} \tag{43}$$

Case 2. SOA rejects V_n.

This implies that the quota-enough condition is not triggered and all the k sub-intervals are accepted by the threshold-accepting condition. Accordingly, we can bound the overall covered length of SOA as follow.

$$Len(\Phi(\mathbb{V}_n)) \geq 1 + (k-1)\theta \tag{44}$$

Suppose $V_i \in \{V_k, ..., V_{n-1}\}$ is the sub-interval that SOA finally accepts, i.e., $|\Phi(\mathbb{V}_i)| = k$ and $Len(\Phi(\mathbb{V}_n)) = Len(\Phi(\mathbb{V}_i))$. In other words, SOA misses all sub-intervals in $\{V_{i+1}, ..., V_n\}$ which could be exploited by OPT. Since SOA misses $n-k$ sub-intervals in total, OPT can get an overall length less than $Len(\Phi(\mathbb{V}_n))+(n-k)m$. Also, OPT cannot get an overall length over km by its quota k in this case. In summary, we bound the overall covered length of OPT as follows.

$$Len(OPT) \leq \min\{km, Len(\Phi(\mathbb{V}_n)) + (n-k)m\} \tag{45}$$

Further, we bound the competitive ratio of SOA under the FL-FN setting as follows.

$$\begin{aligned}
\rho &= \frac{Len(OPT)}{Len(\Phi(\mathbb{V}_n))} \\
&\leq \min\{\frac{km}{1+(k-1)\theta}, 1 + \frac{(n-k)m}{1+(k-1)\theta}\} \\
&\leq \min\{\frac{\sqrt{1+2(k-1)(n-k)m}-1}{k-1} + 1, \frac{\sqrt{(1+8m)k^2-(6+8m)k+9}-k-1}{2(k-1)} + 1\}
\end{aligned} \tag{46}$$

in which the first inequality holds by Inequalities (44) and (45).

By Case 1 and Case 2, the competitive ratio of SOA can be bounded as follows.

$$\rho \leq \max\{1+2\theta, \min\{\frac{km}{1+(k-1)\theta}, 1 + \frac{(n-k)m}{1+(k-1)\theta}\}\} \tag{47}$$

Note that the above upper bound for the FL-FN setting is different from the upper bound (21) for the UL-FN setting as presented in Theorem 4. Intuitively, when SOA accepts k sub-intervals by threshold-accepting condition, OPT possibly exploits more length from those missed sub-intervals in the FL-FN setting than in the UL-FN setting. By applying a trade-off idea similar as that in deriving Equation (35), we get the best-fit threshold θ that minimizes the right-hand side of Inequality (47), which is as follows.

$$\theta = \min\{\frac{\sqrt{1+2(k-1)(n-k)m}-1}{2k-2}, \frac{\sqrt{(1+8m)k^2-(6+8m)k+9}-k-1}{4(k-1)}\} \tag{48}$$

Notice that $\frac{\sqrt{1+2(k-1)(n-k)m}-1}{2k-2}$ decreases in k, while $\frac{\sqrt{(1+8m)k^2-(6+8m)k+9}-k-1}{4(k-1)}$ increases in k, when $k \in \{2, 3, \cdots, n-1\}$. By applying Equation (48) into Inequality (47), we get the upper bound on the competitive ratio of the FL-FN setting as follows.

$$\rho \leq \min\{\frac{\sqrt{1+2(k-1)(n-k)m}-1}{k-1}+1, \frac{\sqrt{(1+8m)k^2-(6+8m)k+9}-k-1}{2(k-1)}+1\} \tag{49}$$

The proof completes. □

Remark 4 Thresholds in (36) and (48) are well-designed so as to achieve a trade-off between the quota and the loss from sub-intervals that are either missed or rejected by SOA, for different settings, respectively. In the case that some slight loss in the upper bound is tolerated, we note that a neat constant upper bound could be guaranteed with some constant thresholds, see the following Corollaries 4, 5 and 6.

Corollary 4 *In the UL-FN setting, SOA with $\theta = 0.5$ guarantees an upper bound of 2 on its competitive ratio.*

Proof According to Theorem 4, the upper bound on the competitive ratio of SOA with threshold $\theta = 0.5$ follows

$$\begin{aligned} &\max\{2, \min\{\frac{2k}{k+1}, 1 + \frac{2(n-k)}{1+k}\}\} \\ &\leq \max\{2, \frac{2k}{k+1}\} \\ &= 2 \end{aligned}$$

The proof completes. □

Corollary 5 *In the US-FN setting, SOA with $\theta = 0.5$ guarantees an upper bound of 2 on its competitive ratio.*

Proof Our proof applies a similar analysis of Theorem 6. Suppose $Len(\Phi(\mathbb{V}_n))$ consists of a number ζ of disjoint intervals. In the case that V_n is accepted by SOA with $\theta = 0.5$, the competitive ratio of SOA could be bounded as follows

$$\begin{aligned} \rho &= \frac{Len(OPT)}{Len(\Phi(\mathbb{V}_n))} \\ &\leq \frac{Len(\Phi(\mathbb{V}_n)) + \min\{2\theta\zeta, \theta(n-k)\}}{Len(\Phi(\mathbb{V}_n))} \\ &\leq 1 + 2\theta = 2 \end{aligned}$$

in which the first inequality holds by Inequality 40 and the second inequality holds because $Len(\Phi(\mathbb{V}_n)) \geq 1$ and $\min\{2\theta\zeta, \theta(n-k)\} \leq 2\theta$. In the case that V_n is not accepted by SOA with $\theta = 0.5$, the competitive ratio of SOA could be bounded as follows

$$\begin{aligned} \rho &= \frac{Len(OPT)}{Len(\Phi(\mathbb{V}_n))} \\ &\leq \frac{\min\{k, Len(\Phi(\mathbb{V}_n)) + (n-k)\}}{Len(\Phi(\mathbb{V}_n))} \\ &\leq \frac{2k}{1+k} \end{aligned}$$

in which the first inequality holds by Inequality (42) and the second inequality holds by Inequality (41). In summary, SOA with $\theta = 0.5$ guarantees an upper bound of $\max\{2, \frac{2k}{1+k}\} = 2$. □

Corollary 6 *In the FL-FN setting, SOA with $\theta = 1$ guarantees an upper bound of $\max\{3, m\}$ on its competitive ratio.*

Proof By applying a similar idea of Theorem 7, we have that the SOA with threshold $\theta = 1$ could achieve an upper bound on its competitive ratio as follows

$$\begin{aligned}\rho &= \frac{Len(OPT)}{Len(\Phi(\mathbb{V}_n))} \\ &\leq \max\{1+2\theta, \min\{\frac{km}{1+(k-1)\theta}, 1+\frac{(n-k)m}{1+(k-1)\theta}\}\} \\ &= \max\{3, m\}\end{aligned}$$

in which the inequality holds by Inequalities (43) and (46). □

Algorithm SOA$_{\text{VN}}$. Now, we formally introduce our algorithm SOA$_{\text{VN}}$ to the VN setting. The SOA$_{\text{VN}}$ remains largely the same as Algorithm 3 of SOA. However, SOA$_{\text{VN}}$ discards the **else if** branch of the quota-enough condition in Lines 6-7 of Algorithm 3, and the threshold θ in Line 9 of Algorithm 3 is updated by Equation (53). Pseudocode of SOA$_{rmVN}$ is given in Algorithm 4.

Algorithm 4 Algorithm SOA$_{\text{VN}}$

Input: A sequence $\mathbb{V} = \{V_1, V_2, ..., \}$ of sub-intervals of the target interval $[0, a]$, in which $V_i = [o_i, d_i]$ for each $V_i \in \mathbb{V}$, the quota k.
Output: A set of accepted sub-intervals, i.e., $\Phi(\mathbb{V})$.

1: $\Phi(\mathbb{V}_1) = \{V_1\}$; ▷ always accept V_1
2: **for** $V_i \in \mathbb{V}$ **do**
3: **if** $|\Phi(\mathbb{V}_{i-1})| = k$ **then**
4: $\Phi(\mathbb{V}) = \Phi(\mathbb{V}_{i-1})$;
5: **break**; ▷ Stop accepting sub-intervals as quota is used up.
6: **else**
7: **if** $Len(\Phi(\mathbb{V}_{i-1}) \cup \{V_i\}) - Len(\Phi(\mathbb{V}_{i-1})) \geq \frac{\sqrt{9k^2-14k+9}-k-1}{4(k-1)}$ **then**
8: $\Phi(\mathbb{V}_i) = \Phi(\mathbb{V}_{i-1}) \cup \{V_i\}$; ▷ accept V_i by threshold-accept
9: **else**
10: $\Phi(\mathbb{V}_i) = \Phi(\mathbb{V}_{i-1})$; ▷ reject V_i
11: **end if**
12: **end if**
13: **end for**

Corollary 7 *With threshold θ given by (53), SOA$_{\text{VN}}$ runs in $O(n)$ time and achieves a competitive ratio no larger than $\frac{\sqrt{9k^2-14k+9}-k-1}{2(k-1)} + 1$ for the UL-VN setting in any limited time frame.*

Proof In the UL-VN setting, where the number $|\mathbb{V}|$ of total released sub-intervals is unknown to the online algorithm in advance, we discuss two cases in any limited time period T.

Case 1. SOA_{VN} runs out of its quota k in time frame T and accepts k sub-intervals by the single threshold $\theta = \frac{\sqrt{9k^2-14k+9}-k-1}{4(k-1)}$.

This implies that the overall covered length by SOA_{VN} is bounded by

$$Len(\Phi(\mathbb{V})) \geq 1 + (k-1)\theta \tag{50}$$

And a trivial upper bound on the covered length of OPT is $Len(OPT) \leq k$. Hence, we have the ratio

$$\rho \leq \frac{k}{1+(k-1)\theta}$$

Case 2. SOA_{VN} still has quota (≥ 1) after the time frame T.

By applying a similar proof idea as in Theorem 5, the competitive ratio of SOA_{rmVN} is bounded as follows

$$\rho \leq 1 + 2\theta \tag{51}$$

By Cases 1-2, the competitive ratio of SOA_{rmVN} can be bounded as follows.

$$\rho \leq \max\{1+2\theta, \frac{k}{1+(k-1)\theta}\} \tag{52}$$

Now, we can find the threshold that minimizes the right-hand side term of Inequality (52) as follows.

$$\theta = \frac{\sqrt{9k^2-14k+9}-k-1}{4(k-1)} \tag{53}$$

By applying Equation (53) to Inequality (52), we have the threshold of SOA_{VN} as

$$\frac{\sqrt{9k^2-14k+9}-k-1}{2(k-1)} + 1 \tag{54}$$

□

Corollary 8 *With threshold θ given by (53), SOA_{VN} runs in $O(n)$ time and achieves a competitive ratio no larger than $\frac{\sqrt{(1+8m)k^2-(6+8m)k+9}-k-1}{2(k-1)} + 1$ for the FL-VN setting in any limited accepting time frame, in which m indicates the maximum length of a sub-interval.*

Proof Since the $n = |\mathbb{V}|$ is discarded in the FL-VN setting, we can upper bound the overall covered length by OPT as km. Further, by a similar proof idea as in Theorem 7, one can derive the upper bound of this corollary. □

5.2 Double-threshold Online Algorithm

Built upon SOA, we present the Double-threshold Online Algorithm (**DOA**) under the FN setting, which remains largely the same as Algorithm 3 but extends the single threshold θ in the threshold-accepting condition to two thresholds θ_1 and θ_2 ($\geq \theta_1$). Specifically, DOA changes its threshold from θ_1 to θ_2 once accepting ω sub-intervals, in which the values of $(\omega, \theta_1, \theta_2)$ are given later by solving the non-linear program (i-viii). Before that, we first give the competitive analysis of DOA. Denote μ as the number of disjoint intervals[14] formed by the sub-intervals accepted by DOA. Then, we have Lemma 1.

[14]Note that the length of each of these μ disjoint intervals could be larger than one as an interval could be formed by multiple accepted sub-intervals.

Lemma 1 *In UL-FN, when DOA accepts ν ($1 \leq \nu \leq k-1$) sub-intervals by the threshold-accept condition and the other $k-\nu$ sub-intervals by the quota-enough condition, OPT can achieve an overall length of at most*

$$\begin{cases} \frac{\min\{k,\mu+(\nu-\mu)\theta_1+2\mu\theta_1\}}{\mu+(\nu-\mu)\theta_1}, & \text{for } 1 \leq \nu \leq \omega-1 \\ \frac{\min\{k,\mu+(\omega-1)\theta_1+(\nu-\mu-\omega+1)\theta_2\}}{\mu+(\omega-1)\theta_1+(\nu-\mu-\omega+1)\theta_2}, & \text{for } \omega \leq \nu \leq k-1 \\ \frac{\min\{k,n-k+1+(\omega-1)\theta_1+(k-\omega)\theta_2\}}{1+(\omega-1)\theta_1+(k-\omega)\theta_2}, & \text{for } \nu = k \end{cases}$$

times the overall length achieved by DOA.

Proof Denote $\{I_1, I_2, \cdots, I_\mu\}$ as the μ disjoint intervals that are formed by those sub-intervals accepted by DOA. Clearly, each of these disjoint intervals has a length no smaller than 1, i.e., $\|I_i\| \geq 1$ for $i \in \{1, \cdots, \mu\}$. Accordingly, we have the overall length achieved by DOA as $Len(\Phi(\mathbb{V}_n)) = \sum_{i=1}^{\mu} \|I_i\|$. Since the coverage problem is only considered in the one-dimensional space (i.e., the target interval $[0, a]$), we care about how much a newly-released sub-interval exceeds either the left or the right side of each of those disjoint intervals. Below we discuss two cases.

Case 1. DOA accepts $\nu \leq \omega - 1$ sub-intervals by threshold.

Recall that each interval of $\{I_1, I_2, \cdots, I_\mu\}$ consists of at least one sub-interval accepted by DOA. We count the overall length $Len(\Phi(\mathbb{V}_n))$ achieved by DOA in the following way. First, we count only one sub-interval in each interval of $\{I_1, I_2, \cdots, I_\mu\}$, implying that $Len(\Phi(\mathbb{V}_n)) \geq \mu$. Note that DOA threshold-accepts sub-intervals only by θ_1 in this case. Then, each of the other accepted sub-interval accumulates a length of θ_1 by DOA. Further, we have

$$Len(\Phi(\mathbb{V}_n)) \geq \mu + (\nu - \mu)\theta_1 \tag{55}$$

Each sub-interval that DOA fails to accept could not exceed either side of an interval in $\{I_1, I_2, \cdots, I_\mu\}$ by θ_1 as otherwise it will also be threshold-accepted by DOA, implying that the overall length accepted by OPT could not be greater than $Len(\Phi(\mathbb{V}_n)) + 2\mu\theta_1$, in which we recall that $Len(\Phi(\mathbb{V}_n))$ denotes overall length accepted by DOA. And clearly, OPT could not accept an overall length larger than k. Hence, we know the ratio of the length accepted by OPT to the length accepted by DOA could not be larger than

$$\frac{\min\{k, Len(\Phi(\mathbb{V}_n)) + 2\mu\theta_1\}}{Len(\Phi(\mathbb{V}_n))} \leq \frac{\min\{k, \mu + (\nu - \mu)\theta_1 + 2\mu\theta_1\}}{\mu + (\nu - \mu)\theta_1}$$

in which the inequality holds by (55)

Case 2. DOA accepts $\omega \leq \nu \leq k - 1$ sub-intervals by threshold.

Among those ν sub-interval, note that DOA accepts the first ω by θ_1 and the last $\nu - \omega$ by θ_2 in this case. Since $\theta_1 \leq \theta_2$ in DOA, we get a lower bound on the overall length achieved by DOA as follows

$$Len(\Phi(\mathbb{V}_n)) \geq \mu + (\omega - 1)\theta_1 + (\nu - \mu - \omega + 1)\theta_2 \tag{56}$$

By a similar analysis as in Case 1, the ratio of the length achieved by OPT to the length achieved by DOA as

$$\frac{\min\{k, Len(\Phi(\mathbb{V}_n)) + 2\mu\theta_1\}}{Len(\Phi(\mathbb{V}_n))} \leq \frac{\min\{k, \mu + (\omega - 1)\theta_1 + (\nu - \mu - \omega + 1)\theta_2\}}{\mu + (\omega - 1)\theta_1 + (\nu - \mu - \omega + 1)\theta_2}$$

in which the inequality holds by (56).

Case 3. DOA accepts k sub-intervals by thresholds.

Since all accepted sub-intervals of DOA are by thresholds θ_1 and θ_2, DOA could achieve an overall length no less than $1+(\omega-1)\theta_1+(k-\omega)\theta_2$ as similarly analyzed in Case 1. In contrast, OPT can fully utilize those sub-intervals missed by DOA but still could not achieve an overall length larger than either $(n-k+1+(\omega-1)\theta_1+(k-\omega)\theta_2)$ or k. In summary, we have the ratio of the length accepted by OPT to the length accepted by DOA could not be larger than

$$\frac{\min\{k, n-k+1+(\omega-1)\theta_1+(k-\omega)\theta_2\}}{1+(\omega-1)\theta_1+(k-\omega)\theta_2}$$

Cases 1, 2, 3 completes the proof of this lemma. □

Theorem 8 *In UL-FN, the competitive ratio of DOA is upper bounded by*

$$\max\{1+2\theta_1, 1+\frac{2\theta_2}{1+\frac{\omega-s}{s}\theta_1}, \frac{k}{s+1+(\omega-s-1)\theta_1}, \frac{\min\{k, n-k+q\}}{q}\} \quad (57)$$

where $q = 1+(\omega-1)\theta_1+(k-\omega)\theta_2$, $s = \frac{k+(1-\omega)\theta_1-2\theta_2}{1-\theta_1+2\theta_2}$.

Proof Recall that DOA threshold-accepts ν sub-intervals and quota-enough accepts the other $(k-\nu)$ sub-intervals, formulating μ disjoint intervals. It could happen that the μ intervals only consist of those threshold-accepted ν sub-intervals, which means those $(k-\nu)$ quota-enough-accepted sub-intervals actually contribute zero. We discuss three cases with regard to the value of ν.

Case 1. $(1 \le \nu \le \omega-1)$. By Lemma 1, the competitive ratio of DOA satisfies (58).

$$\rho \le \frac{\min\{k, Len(\Phi(\mathbb{V}_n))+2\mu\theta_1\}}{Len(\Phi(\mathbb{V}_n))} \le \frac{\min\{k, \mu+(\nu-\mu)\theta_1+2\mu\theta_1\}}{\mu+(\nu-\mu)\theta_1} \le 1+2\theta_1 \quad (58)$$

Case 2. $(\omega \le \nu \le k-1)$. Denote $s+1$ as the minimum number of disjoint intervals formed by those sub-intervals accepted by OPT. Clearly, OPT could achieve a length of at most k, implying the following (59) and (60).

$$s+1+(\omega-s)\theta_1+2(s+1)\theta_2 \ge k \quad (59)$$

$$s+(\omega-s)\theta_1+2s\theta_2 < k \quad (60)$$

We rewrite (59) and (60) as follows

$$\frac{k+(1-\omega)\theta_1-2\theta_2}{1+2\theta_2-\theta_1} \le s \le \frac{k-\omega\theta_1}{1-\theta_1+2\theta_2} \quad (61)$$

Further, we get $s = \frac{k+(1-\omega)\theta_1-2\theta_2}{1+2\theta_2-\theta_1} \in [1, \omega-1]$ satisfying Inequality (61).

Case 2.1. $(\mu \le s)$. By Lemma 1, the ratio is upper bounded by

$$\frac{\mu+(\omega-\mu)\theta_1+(\nu-\omega)\theta_2+2\mu\theta_2}{\mu+(\omega-\mu)\theta_1+(\nu-\omega)\theta_2} \le \frac{s+(\omega-s)\theta_1+2s\theta_2}{s+(\omega-s)\theta_1} = 1+\frac{2\theta_2}{1+\frac{\omega-s}{s}\theta_1} \quad (62)$$

in which the inequality holds by $\mu \le s$ and $\omega \le \nu$.

Case 2.2. $(\mu \ge s+1)$. By Lemma 1, the ratio is upper bounded by

$$\frac{k}{\mu+(\omega-1)\theta_1+(\nu-\mu-\omega+1)\theta_2} \le \frac{k}{s+1+(\omega-s-1)\theta_1} \quad (63)$$

in which the inequality holds by the basic condition $\mu \ge s+1$ of this sub-case.

Case 3. $(\nu = k)$. By Lemma 1, the competitive ratio of DOA is upper bounded by

$$\frac{\min\{k, n-k+1+(\omega-1)\theta_1+(k-\omega)\theta_2\}}{1+(\omega-1)\theta_1+(k-\omega)\theta_2} \quad (64)$$

By Cases 1, 2, 3, the competitive ratio of DOA is upper bounded by (57). □

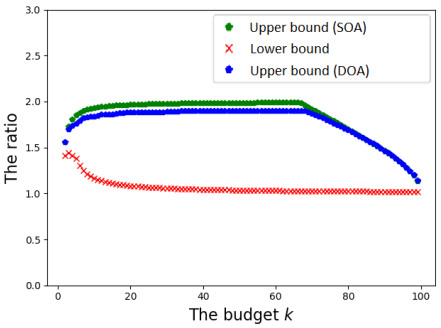

Fig. 2 Performance comparison among DOA, SOA and the lower bound in UL-FN.

To find the parameters $(\omega, \theta_1, \theta_2)$ that minimize the upper bound (i.e., the right-hand side of Equation (57)) of DOA's competitive ratio, we propose the following nonlinear program, in which constraints (ii)-(vi) are transformed from Equation (57) respectively, while constraints (vii) [15] and (viii) are restricted by the model.

$$\min_{(\omega,\theta_1,\theta_2)} \quad C \tag{i}$$

$$\text{s.t.} \quad C \geq 1 + 2\theta_1, \tag{ii}$$

$$C \geq 1 + \frac{2\theta_2}{1 + \frac{\omega - s}{s}\theta_1} \tag{iii}$$

$$C \geq \frac{k}{s + 1 + (\omega - s - 1)\theta_1} \tag{iv}$$

$$C \geq \frac{\min\{k, n - k + 1 + (\omega - 1)\theta_1 + (k - \omega)\theta_2\}}{1 + (\omega - 1)\theta_1 + (k - \omega)\theta_2} \tag{v}$$

$$s = \frac{k + (1 - \omega)\theta_1 - 2\theta_2}{1 + 2\theta_2 - \theta_1} \tag{vi}$$

$$1 \leq \omega \leq k \tag{vii}$$

$$0 < \theta_1 < \theta_2 \leq 1, \tag{viii}$$

Since the program (i)-(viii) is nonlinear and is complicated when transformed into a linear programming, we search its approximated solution under the UL-FN setting by giving the precision of θ as 0.01 and $n = 100$. According to the searching result, we find that the best timing ω is usually around $0.8k$ and $\theta_1 < \theta_2$.With the values $(\omega, \theta_1, \theta_2)$ returned from the program (i)-(viii), DOA slightly improves the performance of SOA, see **Fig. 2**. Regarding the worst-case performances of SOA and DOA, we observe the followings.

[15] Constraint (vii) actually can be restricted, by calculation, to $\left\lceil \frac{k+1}{5} \right\rceil \leq \omega \leq k$.

Note that the two terms $(1+2\theta)$ and $\min\{\frac{k}{1+(k-1)\theta}, 1+\frac{n-k}{1+(k-1)\theta}\}$ in the max operation of SOA's upper bound (see Equation (21) in Theorem 4)) are very close to the two terms $(1+2\theta_2)$ and $\min\{\frac{k}{1+(\omega-1)\theta_1+(k-\omega)\theta_2}, 1+\frac{n-k}{1+(\omega-1)\theta_1+(k-\omega)\theta_2}\}$ in the max operation of DOA's upper bound (see Equation (57) of Theorem 8). Since parameters (θ) and $(\theta_1, \theta_2, \omega)$ seek the trade-off between the quota and algorithm's loss in SOA and DOA, respectively, DOA would not outperform SOA significantly.

It is also worth noting that, when the ratio $\frac{k}{n}$ of the quota over the total number of online sub-intervals is relatively small, we find that more quota induces worse performances of both SOA and DOA. The intuition is that OPT has more chances to gain values from those missed sub-intervals by our algorithms in this case. Conversely, more quota induces better performances of both SOA and DOA when the ratio $\frac{k}{n}$ is relatively large, which is due to that online algorithms have fewer chances to miss length from OPT in this case. The turning point of $\frac{k}{n}$ is around $\frac{2}{3}$ in SOA since the two terms of the competitive ratio in Theorem 5 are monotonically decreasing and increasing, respectively, with regard to k, and meet when $\frac{k}{n} \approx \frac{2}{3}$. Interestingly, the turning point of $\frac{k}{n}$ is also around $\frac{2}{3}$ in DOA (see the example in **Fig. 2**). Particularly, when the thresholds in an algorithm are non-increasing for accepting sub-intervals, we have the following theorem.

Theorem 9 *SOA outperforms any online deterministic algorithm that accepts sub-intervals by non-increasing thresholds.*

Proof Denote ALG as an online algorithm with non-increasing thresholds that are concerned in this theorem. We discuss the competitive ratio of ALG in the UL setting in a limited time frame of T, and we abuse notations to denote $Len(\Phi(\mathbb{V}))$ as the overall length achieved by ALG in T. Before our discussion, we define ALG as follows: ALG uses threshold θ_i to accept its ith sub-interval (in which the thresholds satisfy $\theta_1 \geq \theta_2 \cdots \geq \theta_k$) and terminates when either the quota k or the time frame T runs out. Suppose, w.l.o.g., that ALG finally accepts y sub-intervals within the given time frame T, formulating x disjoint intervals of the target interval $[0, a]$. Now, we discuss k cases with regard to y, in each of which we further discuss y sub-cases with regard to x.

Case 1. $(y = 1)$. Clearly, we have $x = 1$. Note that $Len(OPT)$ cannot exceed $Len(\Phi(\mathbb{V}))$ by $2\theta_2$ as otherwise ALG can accept another sub-interval by $\theta_2 \leq \theta_1$, we have

$$\rho_{\text{case 1}} = \frac{1+2\theta_2}{1} = 1+2\theta_2.$$

Case 2. $(y = 2)$. We have two sub-cases.

- **Case 2.1.** $(x = 1)$. We have

$$Len(\Phi(\mathbb{V})) \geq 1+\theta_2$$

and

$$Len(OPT) \leq 1+\theta_2+2\theta_3$$

as otherwise ALG can accept the third sub-interval by threshold $\theta_3 \leq \theta_2$. Hence, the ratio can be bounded as follows

$$rho_{\text{case 2.1}} = \frac{Len(OPT)}{Len(\Phi(\mathbb{V}))} = \frac{1+\theta_2+2\theta_3}{1+\theta_2} = 1 + \frac{2\theta_3}{1+\theta_2}.$$

- **Case 2.2.** ($x = 2$). We have $Len(\Phi(\mathbb{V})) = 2$ in this UL setting and
$$Len(OPT) \leq 2 + 4\theta_3$$
as otherwise ALG can accept the third sub-interval by threshold $\theta_3 \leq \theta_2$. Hence, the ratio of this subcase can be bounded by
$$\rho_{\text{case 2.2}} = \frac{Len(OPT)}{Len(\Phi(\mathbb{V}))} \leq \frac{2+4\theta_3}{2} = 1 + 2\theta_3$$
which is larger than $\rho_{\text{case 2.1}} = 1 + \frac{2\theta_3}{1+\theta_2}$.

Overall, the ratio of Case 2 is upper bounded by

$$\rho_{\text{case 2.2}} = 1 + 2\theta_3 < 1 + \theta_2.$$

Case $[\frac{k}{2}]$. ($y = [\frac{k}{2}]$). We have $[\frac{k}{2}]$ subcases.

- **Case $[\frac{k}{2}]$.1.** ($x = 1$). We have $Len(\Phi(\mathbb{V})) \geq 1 + \sum_{i=2}^{[\frac{k}{2}]} \theta_i$ and
$$Len(OPT) \leq Len(\Phi(\mathbb{V})) + 2\theta_{[\frac{k}{2}]+1} = 1 + \sum_{i=2}^{[\frac{k}{2}]} \theta_i + 2\theta_{[\frac{k}{2}]+1}$$
in which the inequality holds as otherwise ALG can accept the $([\frac{k}{2}]+1)$th sub-interval by threshold $\theta_{[\frac{k}{2}]+1} \leq \theta_{[\frac{k}{2}]}$. Hence,
$$\rho_{\text{case } [\frac{k}{2}].1.} = \frac{Len(OPT)}{Len(\Phi(\mathbb{V}))} \leq 1 + \frac{2\theta_{[\frac{k}{2}]+1}}{1+\sum_{i=2}^{[\frac{k}{2}]}} \theta_i$$

- **Case $[\frac{k}{2}]$.2.** ($x = 2$). We have $Len(\Phi(\mathbb{V})) \geq 2 + \sum_{i=3}^{[\frac{k}{2}]} \theta_i$ and
$$Len(OPT) \leq Len(\Phi(\mathbb{V})) + 4\theta_{[\frac{k}{2}]+1} = 2 + \sum_{i=3}^{[\frac{k}{2}]} \theta_i + 4\theta_{[\frac{k}{2}]+1}$$
in which the inequality holds as otherwise ALG can accept the $([\frac{k}{2}]+1)$th sub-interval by threshold $\theta_{[\frac{k}{2}]+1} \leq \theta_{[\frac{k}{2}]}$. Hence,
$$\rho_{\text{case } [\frac{k}{2}].2.} = \frac{Len(OPT)}{Len(\Phi(\mathbb{V}))} \leq 1 + \frac{4\theta_{[\frac{k}{2}]+1}}{2+\sum_{i=3}^{[\frac{k}{2}]} \theta_i} = 1 + \frac{2\theta_{[\frac{k}{2}]+1}}{1+\frac{\sum_{i=3}^{[\frac{k}{2}]} \theta_i}{2}}$$

- **Case $[\frac{k}{2}]$.j-1.** ($x = j-1$)[16]. We have $Len(\Phi(\mathbb{V})) \geq j - 1 + \sum_{i=j}^{[\frac{k}{2}]} \theta_i$ and
$$Len(OPT) \leq Len(\Phi(\mathbb{V})) + 2(j-1)\theta_{[\frac{k}{2}]+1}$$
in which the inequality holds as otherwise ALG can accept the $([\frac{k}{2}]+1)$th sub-interval by threshold $\theta_{[\frac{k}{2}]+1} \leq \theta_{[\frac{k}{2}]}$. Hence,
$$\rho_{\text{case } [\frac{k}{2}].j-1.} = \frac{Len(OPT)}{Len(\Phi(\mathbb{V}))} \leq 1 + \frac{2(j-1)\theta_{[\frac{k}{2}]+1}}{j-1+\sum_{i=j}^{[\frac{k}{2}]} \theta_i} = 1 + \frac{2\theta_{[\frac{k}{2}]+1}}{1+\frac{\sum_{i=3}^{[\frac{k}{2}]} \theta_i}{j-1}}$$

Note that $\rho_{\text{case } [\frac{k}{2}].j-1.} \geq \rho_{\text{case } [\frac{k}{2}].1.} = 1 + \frac{2\theta_{[\frac{k}{2}]+1}}{1+\sum_{i=3}^{[\frac{k}{2}]} \theta_i}$.

[16] Here, the j is chosen such that $2j + 2 \leq k < 3j$.

- **Case** $[\frac{k}{2}]$**.j.** $(x = j)$. We have $Len(\Phi(\mathbb{V})) \geq j + \sum_{i=j+1}^{[\frac{k}{2}]} \theta_i$ and
$$Len(OPT) \leq Len(\Phi(\mathbb{V})) + (k-j)\theta_{[\frac{k}{2}]+1}$$
in which the inequality holds as OPT can exceed ALG by at most $(k-j)\theta_{[\frac{k}{2}]+1}$ by $k < 3j$. Hence,
$$\rho_{\text{case } [\frac{k}{2}].j.} = \frac{Len(OPT)}{Len(\Phi(\mathbb{V}))} \leq 1 + \frac{(k-j)\theta_{[\frac{k}{2}]+1}}{j + \sum_{i=j+1}^{[\frac{k}{2}]} \theta_i} = 1 + \frac{2\theta_{[\frac{k}{2}]+1}}{\frac{2j}{k-j} + \frac{2\sum_{i=j+1}^{[\frac{k}{2}]} \theta_i}{k-j}}$$
Note that $\rho_{\text{case } [\frac{k}{2}].j.} \leq \rho_{\text{case } [\frac{k}{2}].1.} = 1 + \frac{2\theta_{[\frac{k}{2}]+1}}{1+\sum_{i=3}^{[\frac{k}{2}]} \theta_i} \leq \rho_{\text{case } [\frac{k}{2}].j-1.}$, in which the first inequality holds naturally since $k \leq 3k$.

- **Case** $[\frac{k}{2}]$**.j+1.** $(x = j+1)$. We have $Len(\Phi(\mathbb{V})) \geq j + 1 + \sum_{i=j+2}^{[\frac{k}{2}]} \theta_i$ and
$$Len(OPT) \leq Len(\Phi(\mathbb{V})) + k\theta_{[\frac{k}{2}]+1}$$
in which the inequality holds as OPT can exceed ALG by at most $k\theta_{[\frac{k}{2}]+1}$ since $k < 3j$. Hence,
$$\rho_{\text{case } [\frac{k}{2}].j+1.} = \frac{Len(OPT)}{Len(\Phi(\mathbb{V}))} \leq 1 + \frac{k\theta_{[\frac{k}{2}]+1}}{j + 1 + \sum_{i=j+2}^{[\frac{k}{2}]} \theta_i}$$
note that $\rho_{\text{case } [\frac{k}{2}].j+1.} \leq \rho_{\text{case } [\frac{k}{2}].j.}$ as $\theta \leq 1$.

We note that the ratios of the sub-cases above in Case $[\frac{k}{2}]$ increase as x increases within $\{1, \cdots, j-1\}$ and further decrease as x increases within $\{j, \cdots, y\}$. Overall, we get the ratio of this case as
$$\rho_{\text{case}[\frac{k}{2}]} = \max_{1 \leq x \leq [\frac{k}{2}]} \{\rho_{\text{case}[\frac{k}{2}].\text{x}}\} = \rho_{\text{case}[\frac{k}{2}].\text{j}-1} = 1 + \frac{2\theta_{[\frac{k}{2}]+1}}{1 + \frac{\sum_{i=3}^{[\frac{k}{2}]} \theta_i}{j-1}}$$
$$< 1 + 2\theta_{[\frac{k}{2}]+1} \leq 1 + 2\theta_2 = \rho_{\text{case1}}.$$

Case k**.** $(y = k)$. Clearly, $Len(OPT) \leq k$ while $Len(\Phi(\mathbb{V})) \geq 1 + \theta_1 + \cdots + \theta_k$. Hence, we have
$$\rho_{\text{case k}} \leq \frac{k}{1 + \sum_{i=2}^{k} \theta_i}.$$

Notice that the ratios of the cases above decrease as y increases till $y = k-1$. Therefore, the competitive ratio of ALG is upper bounded by
$$\begin{aligned}
&\max\{\frac{k}{1 + \sum_{i=2}^{k} \theta_i}, 1 + 2\theta_2\} \\
&\geq \max\{\frac{k}{1 + (k-1)\theta_2}, 1 + 2\theta_2\} \\
&\geq \underbrace{\frac{\sqrt{9k^2 - 14k + 9} - k - 1}{2(k-1)} + 1}_{\text{the ratio of SOA in VN}} \\
&\geq \underbrace{\min\{\frac{\sqrt{1 + 2(k-1)(n-k)} - 1}{k-1} + 1, \frac{\sqrt{9k^2 - 14k + 9} - k - 1}{2(k-1)} + 1\}}_{\text{the ratio of SOA in FN}}
\end{aligned}$$

in which the first inequality holds by $\theta_1 \geq \theta_2 \cdots \geq \theta_k$, and the second inequality holds by

$$\frac{\sqrt{9k^2-14k+9}-k-1}{4(k-1)} = \min_{\theta_2} \max\{\frac{k}{1+(k-1)\theta_2}, 1+2\theta_2\}.$$

The proof completes. □

Bound gap analysis. The main results of this paper are summarized in Table 2, in which some complicated parameter-dependent results are approximated, for ease of understanding, by formulations in bold. We refer interested readers to the corresponding theorems and corollaries for accurate results. From Table 2, one can observe that gap between our lower bound (LB) and upper bound (UB) is small in either UL or US setting. For example, LB and UB of UL-VN depend on the quota k, in which LB equals $\sqrt{2} \approx 1.4142$ for $k = 2$ and decreases to its limit one as k increases, while UB is around 1.5616 for $k = 2$ and approaches its limit two as k increases. As such, the bound gap of UL-VN increases from around $1.5616 - 1.4141 = 0.1475$ to no more than one as k increases from 2. In UL-FN and US-FN settings, the bound gap even narrows, this is because UB of UL-FN and US-FN settings takes a min operation of the UB of UL-VN and $(\frac{\sqrt{1+2(k-1)(n-k)}-1}{k-1} + 1)$ which decreases as k increases. *Insights on online recruiting practice*, due to our bound gap analysis, we know that fewer quotas could induce better worst-case performance by our algorithm in the sense that the UB approaches LB more closely.

In the FL setting, the bound gap, which depends on the quota k and the maximum length m of a sub-interval, is a bit more intricate to analyze. The bound gap of UB-LB follows

$$\text{UB} - \text{LB} \leq \begin{cases} \frac{k}{k-1}\frac{\sqrt{1+8m}}{2} + \frac{2}{m+1} - 1, & \text{when } n \geq 2k; \\ 1 + \frac{k\sqrt{1+8m}}{2(k-1)} - \frac{2km}{3km-mn+n-k}, & \text{when } n < 2k. \end{cases} \tag{65}$$

By seeking the first-order derivative with respect to m on the right-hand side terms of Inequality (65), we know that the bound gap is around two for $m = 2$ and generally increases as m increases, which implies that the minor derivation among online workers could better our algorithms' worst-case performance.

6 Concluding Remarks

This paper studies the online maximum k-Interval coverage problem. With regard to the length of each sub-interval and the total number of sub-intervals released, we comprehensively investigate different settings. Our contribution is three-fold.

To shed light on our online algorithm design and bound analysis, we *first* investigate the offline problem from the perspective of a dynamic programming approach. Accordingly, we propose two polynomial-time optimal solutions to the offline problem. By an observed feature of an optimal solution, we find

Table 2 Main results of this work.

Settings		Lower bounds	Upper bounds
UL	FN	$\sqrt{2}$ for $k = 2$ decrease as $k \geq 3$ increase (see Theorem 2)	$\mathbf{< 2}$ **(see Theorem 5)**
	VN	$\sqrt{2}$ for $k = 2$ decrease as $k \geq 3$ increase (see Corollary 1)	$\frac{\sqrt{9k^2-14k+9}-k-1}{2(k-1)} + 1$ (see Corollary 7)
FL	FN	$\frac{2km}{2km-(m-1)\min\{k,n-k\}} (< 2)$ (see Theorem 3)	$\mathbf{< 1 + \frac{k}{k-1}\sqrt{\frac{1+8m}{4}}}$ **(see Theorem 7)**
	VN	$\frac{2m}{m+1}$ (Corollary 2)	$\frac{\sqrt{(1+8m)k^2-(6+8m)k+9}-k-1}{2(k-1)} + 1$ (Corollary 8)
AL	FN or VN	$+\infty$ (see Theorem 1)	-
US	FN	$\sqrt{2}$ for $k = 2$ decrease as $k \geq 3$ increase (see Corollary 3)	$\mathbf{< 2}$ **(see Theorem 6)**

that a dynamic programming-based solution could be accelerated. *Then,* we present lower bounds on the competitive ratio for the online problem under different settings, which are based on our well-designed release scheme of intervals. *Further*, we present two $O(n)$-time online algorithms, including a single-threshold-based algorithm SOA and a double-threshold-based algorithm DOA. DOA uses its first threshold (which is usually set below 0.5) for exploration in accepting the first $[0.8k]$ sub-intervals released and uses its second threshold (which is set to be larger than the first threshold) for exploitation in accepting the remaining $k - [0.8k]$ sub-intervals. We prove that SOA achieves competitive ratios close to the lower bounds in those settings, respectively, and DOA, with its parameters computed by our proposed program, slightly outperforms SOA. In addition, we show that more thresholds could not induce better worst-case performance if those thresholds follow a non-decreasing order in accepting sub-intervals.

One interesting direction of the future work is to consider the case that sub-intervals may have different lengths, which has practical motivation in crowding-sourcing activities where online workers have different expertise ranges.

Statements and Declarations

- **Competing interests.** The authors have no competing interests to declare that are relevant to the content of this article.
- **Availability of data and materials.** Data sharing is not applicable to this article as no datasets were generated or analysed during the current study.

References

Albers S, Ladewig L (2021) New results for the k-secretary problem. Theoretical Computer Science 863:102–119. https://doi.org/10.1016/j.tcs.2021.02.022

Alon N, Awerbuch B, Azar Y, et al (2009) The online set cover problem. SIAM Journal on Computing 39(2):361–370. https://doi.org/10.1137/060661946

Assadi S, Khanna S, Li Y (2019) Tight bounds for single-pass streaming complexity of the set cover problem. SIAM Journal on Computing 50(3):16–341. https://doi.org/10.1137/16M1095482

Ausiello G, Boria N, Giannakos A, et al (2012) Online maximum k-coverage. Discrete Applied Mathematics 160(13-14):1901–1913

Babaioff M, Immorlica N, Kempe D, et al (2007) A knapsack secretary problem with applications. In: Approximation, randomization, and combinatorial optimization. Algorithms and techniques. Springer, p 16–28, https://doi.org/10.1007/978-3-540-74208-1_2

Bateni M, Hajiaghayi M, Zadimoghaddam M (2013) Submodular secretary problem and extensions. ACM Transactions on Algorithms 9(4):1–23. https://doi.org/10.1145/2500121

Brawley AM, Pury CL (2016) Work experiences on mturk: Job satisfaction, turnover, and information sharing. Computers in Human Behavior 54:531—-546. https://doi.org/10.1016/j.chb.2015.08.031

Buchbinder N, Feldman S, Schwartz R (2019) Online submodular maximization with preemption. ACM Transactions on Algorithms 15(3):1—-31. https://doi.org/10.1145/3309764

Chin FY, Chrobak M, Fung SP, et al (2006) Online competitive algorithms for maximizing weighted throughput of unit jobs. Journal of Discrete Algorithms 4(2):255–276. https://doi.org/10.1016/j.jda.2005.03.005

Chrobak M, Jawor W, Sgall J, et al (2007) Online scheduling of equal-length jobs: Randomization and restarts help. SIAM Journal on Computing 36(6):1709–1728. https://doi.org/10.1137/S0097539704446608

Cohen R, Gonen M (2019) On interval and circular-arc covering problems. Annals of Operations Research 275(2):281–295. https://doi.org/10.1007/s10479-018-3025-6

Feldman M, Zenklusen R (2018) The submodular secretary problem goes linear. SIAM Journal on Computing 47(2):330–366. https://doi.org/10.1137/16M1105220

Feldman M, Svensson O, Zenklusen R (2018) A framework for the secretary problem on the intersection of matroids. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, pp 735–752, https://doi.org/10.1137/1.9781611975031.48

Hochbaum DS, Pathria A (1998) Analysis of the greedy approach in problems of maximum k-coverage. Naval Research Logistics (NRL) 45(6):615–627. https://doi.org/10.1002/(SICI)1520-6750(199809)45:6⟨615::AID-NAV5⟩3.0.CO;2-5

Iwama K, Taketomi S (2002) Removable online knapsack problems. In: International Colloquium on Automata, Languages, and Programming, Springer, pp 293–305, https://doi.org/10.1007/3-540-45465-9_26

Khuller S, Moss A, Naor JS (1999) The budgeted maximum coverage problem. Information processing letters 70(1):39–45. https://doi.org/10.1016/S0020-0190(99)00031-9

Klee V (1977) Can the measure of be computed in less than o (n log n) steps? The American Mathematical Monthly 84(4):284–285

Kleinberg R (2005) A multiple-choice secretary algorithm with applications to online auctions. In Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms

Li L, Wei Z, Hao J, et al (2021) Probability learning based tabu search for the budgeted maximum coverage problem. Expert Systems With Applications 183(115310):16–341. https://doi.org/10.1016/j.eswa.2021.115310

Li S, Li M, Duan L, et al (2020) Online maximum k-interval coverage problem. In: International Conference on Combinatorial Optimization and Applications, Springer, pp 455–470, https://doi.org/10.1007/978-3-540-74208-1_2

Rawitz D, Rosén A (2021) Online budgeted maximum coverage. Algorithmica 83(9):2989–3014. https://doi.org/10.1007/s00453-021-00850-7

Saha B, Getoor L (2009) On maximum coverage in the streaming model & application to multi-topic blog-watch. In: Proceedings of the 2009 siam international conference on data mining, SIAM, pp 697–708, https://doi.org/10.1137/1.9781611972795.60

Vaze R (2018) Online knapsack problem under expected capacity constraint. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, IEEE, pp 2159–2167, https://doi.org/10.1109/INFOCOM.2018.8485980