# Incremental Maximization via Continuization

**Yann Disser**
TU Darmstadt, Germany

**Max Klimm**
TU Berlin, Germany

**Kevin Schewior**
University of Southern Denmark, Odense, Denmark

**David Weckbecker**
TU Darmstadt, Germany

## Abstract

We consider the problem of finding an incremental solution to a cardinality-constrained maximization problem that not only captures the solution for a fixed cardinality, but also describes how to gradually grow the solution as the cardinality bound increases. The goal is to find an incremental solution that guarantees a good competitive ratio against the optimum solution for all cardinalities simultaneously. The central challenge is to characterize maximization problems where this is possible, and to determine the best-possible competitive ratio that can be attained. A lower bound of 2.18 and an upper bound of $\varphi + 1 \approx 2.618$ are known on the competitive ratio for monotone and accountable objectives [Bernstein et al., Math. Prog., 2022], which capture a wide range of maximization problems. We introduce a continuization technique and identify an optimal incremental algorithm that provides strong evidence that $\varphi + 1$ is the best-possible competitive ratio. Using this continuization, we obtain an improved lower bound of 2.246 by studying a particular recurrence relation whose characteristic polynomial has complex roots exactly beyond the lower bound. Based on the optimal continuous algorithm combined with a scaling approach, we also provide a 1.772-competitive randomized algorithm. We complement this by a randomized lower bound of 1.447 via Yao's principle.

**2012 ACM Subject Classification** Theory of computation → Online algorithms; Mathematics of computing → Combinatorial algorithms; Mathematics of computing → Combinatorial optimization

**Keywords and phrases** incremental optimization, competitive analysis, robust matching, submodular function

**Digital Object Identifier** 10.4230/LIPIcs.ICALP.2023.47

**Category** Track A: Algorithms, Complexity and Games

**Related Version** *Full Version*: `https://arxiv.org/abs/2305.01310` [5]

**Funding** *Max Klimm*: Supported by Deutsche Forschungsgemeinschaft under Germany's Excellence Strategy, Berlin Mathematics Research Center (grant EXC-2046/1, Project 390685689).
*Kevin Schewior*: Supported in part by the Independent Research Fund Denmark, Natural Sciences, grant DFF-0135-00018B.
*David Weckbecker*: Supported by DFG grant DI 2041/2.

## 1 Introduction

A classical optimization problem takes as input a single instance and outputs a single solution. While this paradigm can be appropriate in static situations, it fails to capture scenarios that are characterized by perpetual growth, such as growing infrastructure networks, expanding companies, or private households with a steady income. In these cases, a single static solution may be rendered useless unless it can be extended perpetually into larger, more expansive solutions that are adequate for the changed circumstances.

© Yann Disser, Max Klimm, Kevin Schewior, and David Weckbecker;
licensed under Creative Commons License CC-BY 4.0

To capture scenarios like this more adequately, we adopt the *incremental optimization* framework formalized as follows. An instance of the Incremental Maximization (IncMax) problem is given by a countable set $U$ of elements and a monotone objective function $f\colon 2^U \to \mathbb{R}_{\geq 0}$ that assigns each subset $X \subseteq U$ a value $f(X)$. A solution for an IncMax instance is an order $\sigma = (e_1, e_2, \dots)$ of the elements of $U$ such that each prefix of $\sigma$ yields a good solution with respect to the objective function $f$. Formally, for $k \in [n]$, let $\textsc{Opt}(k) = \max\{f(X) : |X| = k, X \subseteq U\}$ denote the optimal value of the problem of maximizing $f(X)$ under the cardinality-constraint $|X| = k$. A deterministic solution $\sigma = (e_1, e_2, \dots)$ is called $\alpha$-competitive if $\textsc{Opt}(k)/f(\{e_1, \dots, e_k\}) \leq \alpha$ for all $k \in [n]$. A randomized solution is a probability distribution $\Sigma = (E_1, E_2, \dots)$ over deterministic solutions (where $E_1, E_2, \dots$ are random variables). It is called $\alpha$-competitive if $\textsc{Opt}(k)/\mathbb{E}[f(\{E_1, \dots, E_k\})] \leq \alpha$ for all $k \in [n]$. In both cases, we call the infimum over all $\alpha \geq 1$, such that the solution is $\alpha$-competitive, the *(randomized) competitive ratio* of the solution. A (randomized) algorithm is called $\alpha$-competitive for some $\alpha \geq 1$ if, for every instance, it produces an $\alpha$-competitive solution, and its *(randomized) competitive ratio* is the infimum over all such $\alpha$. The *(randomized) competitive ratio* of a class of problems (or a problem instance) is the infimum over the competitive ratios of all (randomized) algorithms for it.

Clearly, in this general form, no meaningful results regarding the existence of competitive solutions are possible. For illustration consider the instance $U = \{a, b, c\}$ where for some $M \in \mathbb{N}$ we have

$$f(X) = \begin{cases} M, & \text{if } \{b,c\} \subseteq X, \\ |\{a\} \cap X|, & \text{otherwise} \end{cases} \qquad \text{for all } X \subseteq U.$$

Then, every solution needs to start with element $a$ in order to be competitive for $k = 1$, but any such order cannot be better than $M$-competitive for $k = 2$. The underlying issue is that the optimal solution for $k = 2$ given by $\{b, c\}$ does not admit a competitive partial solution of cardinality $k = 1$. To circumvent this issue, Bernstein et al. [1] consider *accountable* functions, i.e., functions $f$, such that, for every $X \subseteq U$, there exists $e \in X$ with $f(X \setminus \{e\}) \geq f(X) - f(X)/|X|$. They further show that many natural incremental optimization problems are monotone and accountable such as the following.

**Weighted matching:** $U$ is the set of edges of a weighted graph, and $f(X)$ is the maximum weight of a matching contained in $X$;

**Set packing:** $U$ is a set of weighted subsets of a ground set, and $f(X)$ is the maximum weight of a set of mutually disjoint subsets of $X$;

**Submodular function maximization:** $U$ is arbitrary, and $f$ is monotone and submodular;

**(Multi-dimensional) Knapsack:** $U$ is a set of items with (multi-dimensional) sizes and values, and $f(X)$ is the maximum value of a subset of items of $X$ that fits into the knapsack.

Bernstein et al. [1] gave an algorithm to compute a $(1 + \varphi)$-competitive incremental solution and showed that the competitive ratio of the IncMax problem is at least 2.18. Throughout this work, we assume that the objective $f$ is accountable.

**Our results.** As a first step, we reduce the general IncMax problem to the special case of IncMaxSep, where the elements of the instance can be partitioned into a (countable) set of uniform and modular subsets such that the overall objective is the maximum over the modular functions on the subsets. We then define the IncMaxCont problem as a continuization, where there exists one such subset with (fractional) elements of every size $c \in \mathbb{R}_{>0}$. The smooth structure of this problem better lends itself to analysis.

We consider the continuous algorithm GreedyScaling($c_1, \rho$) that adds a sequence of these subsets, starting with the subset of size $c_1 > 0$ and proceeding along a sequence of subsets of largest possible sizes under the constraint that $\rho$-competitiveness is maintained for as long as possible. We first show that there always exists an optimal solution of this form.

▶ **Theorem 1.** *For every instance of IncMaxCont, there exists a starting value $c_1$ such that the algorithm GreedyScaling($c_1, \rho^*$) achieves the best-possible competitive ratio $\rho^* \geq 1$.*

Our continuous embedding allows us to view every algorithm as an increasing sequence of sizes of subsets that are added one after the other. Using elementary calculus, we can show that, with the golden ratio $\varphi := \frac{1}{2}(1+\sqrt{5}) \approx 1.618$, GreedyScaling($c_1, \rho$) achieves the known upper bound of $\varphi + 1$ for a range of starting values. Here, $d(c)$ refers to the density, i.e., value per size, of the subset of size $c$ (see Sec. 2).

▶ **Theorem 2.** *GreedyScaling($c_1, \varphi + 1$) is ($\varphi + 1$)-competitive if and only if $d(c_1) \geq \frac{1}{\varphi+1}$.*

On the other hand, we are able to, for every starting value $c_1$, construct an instance of IncMaxCont where GreedyScaling($c_1, \rho$) is not better than ($\varphi + 1$)-competitive for any $\rho > 1$. We emphasize that the optimum value of $\varphi + 1$ emerges naturally from the geometry of complex roots. Based on this evidence, we conjecture that $\varphi + 1$ is the best-possible competitive ratio.

Of course, proving a general lower bound requires to construct a single instance such that GreedyScaling is not better than ($\varphi + 1$)-competitive for *every* starting value. Careful chaining of our construction for a single starting value yields the following.

▶ **Proposition 3.** *For every countable set $S \subset \mathbb{R}_{>0}$ of starting values, there exists an instance of IncMaxCont such that GreedyScaling($c_1, \rho$) is not $\rho$-competitive for any $c_1 \in S$ and any $\rho < \varphi + 1$.*

Crucially, while this gives a lower bound if we only allow rational starting values $c_1 \in \mathbb{Q}$, transferring the lower bound back to IncMax requires excluding all reals. Even though we are not able to achieve this, we can extrapolate our analysis in terms of complex calculus to any IncMaxCont algorithm. With this, we beat the currently best known lower bound of 2.18 in [1].

▶ **Theorem 4.** *The IncMax problem has a competitive ratio of at least 2.246.*

We can also apply our technique, specifically the reduction to separable problem instances and the structure of the GreedyScaling algorithm, to the analysis of randomized algorithms for IncMax. We employ a scaling approach based on the algorithms in [1], combined with a randomized selection of the starting value $c_1$ inspired by a randomized algorithm for the CowPath problem in [16]. The resulting algorithm has a randomized competitive ratio that beats our deterministic lower bound.

▶ **Theorem 5.** *IncMax admits a 1.772-competitive randomized algorithm.*

We complement this result with a lower bound via Yao's principle for separable instances of IncMax.

▶ **Theorem 6.** *Every randomized IncMax algorithm has competitive ratio at least 1.447.*

**Related work.** Our work is based on the incremental maximization framework introduced by Bernstein et al. [1]. We provide a new structural understanding that leads to a better lower bound and new randomized bounds.

A similar framework is considered for matchings by Hassin and Rubinstein [13]. Here, the objective $f$ is the total weight of a set of edges and the solution is additionally required to be a matching. Hassin and Rubinstein [13] show that the competitive ratio in this setting is $\sqrt{2}$ and Matuschke, Skutella, and Soto [19] show that the randomized competitive ratio is $\ln(4) \approx 1.38$. The setting was later generalized to the intersection of matroids [7] and to independence systems with bounded exchangeability [15, 21]. Note that, while our results hold for a broader class of objective functions, we require monotonicity of the objective and cannot model the constraint that the solution must be a matching. We can, however, capture the matching problem by letting the objective $f$ be the largest weight of a matching contained as a subset in the solution (i.e., not all parts of the solution need to be used). That being said, it is easy to verify that the lower bound of $\sqrt{2}$ on the competitiveness of any deterministic algorithm in the setting of [13] also applies in our case.

Hassin and Segev [14] studied the problem of finding a small subgraph that contains, for all $k$, a path (or tree) of cardinality at most $k$ with weight at least $\alpha$ times the optimal solution and show that for this $\alpha|V|/(1-\alpha^2)$ edges suffice. There are further results on problems where the items have sizes and the cardinality-constraint is replaced by a knapsack constraint [4, 6, 17, 20]. Goemans and Unda [9] studied general incremental maximization problems with a sum-objective.

Incremental *minimization* problems further been studied for a variety of minimzation problems such as $k$-median [3, 22, 18], facility location [18, 23], and $k$-center [10, 18]. As noted by Lin et al. [18], the results for the minimum latency problem in [2, 8] implicitly yield results for the incremental $k$-MST problem. There are further results on incremental optimization problems where in each step the set of feasible solution increases [11, 12].

## 2 Separability of Incremental Maximization

As a first step to bound the competitive ratio of IncMax, we introduce a subclass of instances of a relatively simple structure, and show that it has the same competitive ratio as IncMax. Thus, we can restrict ourselves to this subclass in our search for bounds on the competitive ratio.

▶ **Definition 7.** *An instance of* IncMax *with objective $f\colon 2^U \to \mathbb{R}_{>0}$ is called* separable *if there exist a partition $U = R_1 \cup R_2 \cup \dots$ of $U$ and values $d_i > 0$ such that*

$$f(X) = \max_{i\in\mathbb{N}}\{|X \cap R_i| \cdot d_i\} \quad \text{for all } X \subseteq U.$$

*We refer to $d_i$ as the* density *of set $R_i$ and to $v_i := |R_i| \cdot d_i$ as the* value *of set $R_i$. The restriction of* IncMax *to separable instances will be denoted by* IncMaxSep.

We start our analysis of IncMaxSep with the following immediate observation.

▶ **Lemma 8.** *Any instance of* IncMaxSep *can be transformed into one with the same or a worse competitive, that satisfies the following properties.*

1. *There is exactly one set of every cardinality, i.e., $|R_i| = i$.*
2. *Densities are decreasing, i.e., $1 \geq d_1 \geq d_2 \geq \dots$.*
3. *Values are increasing, i.e., $v_1 \leq v_2 \leq \dots$.*

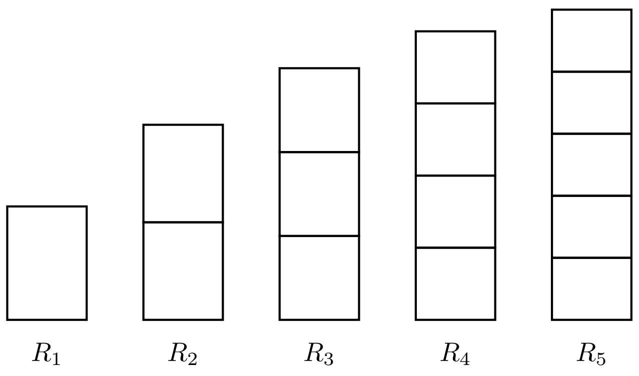

**Figure 1** Illustration of an instance of IncMaxSep with $N = 5$ sets. Each set $R_i$ consists of $i$ elements. The height of the elements represents their value. As in Lemma 8, the values of the single elements becomes less the larger $i$ is, while the value of the whole set $R_i$ increases.

**Proof.** We will show that every instance that does not satisfy the assumptions can be transformed into one that does, without changing the optimum value for any size, and without changing the value of the best incremental solution. Thus the competitive ratio of the two instances coincide.

If there are two sets $R_i, R_j$ with $|R_i| = |R_j|$, it only makes sense to consider the one with higher density, as every solution adding elements from the set of lower density can be improved by adding elements from the other set instead. If there is $i \in \mathbb{N}_{\geq 2}$ such that there is no set with $i$ elements, we can add a new set with $i$ elements to the instance. This new set will have value $v_{i-1}$ . Then, every solution that adds elements from the newly introduced set can be improved by adding elements from set $R_{i-1}$ instead. If there is no set $R_1$ with 1 element, we can introduce it with density $d_2$. Then, every solution that adds this one element can instead also add one element from $R_2$. Thus, the first assumption can be made.

The assumption that $1 \geq d_1$ is without loss of generality by rescaling the objective $f$. If there was $i \in \mathbb{N}$ with $d_i < d_{i+1}$, every solution to the problem instance that adds elements from the set $R_i$ could be improved by adding elements from the set $R_{i+1}$ instead. Since $|R_{i+1}| \geq |R_i|$, this is possible.

The third assumption can be made because, if there was $i \in \mathbb{N}$ with $v_i > v_{i+1}$, a solution that adds elements from $R_{i+1}$ can be improved by adding elements from $R_i$ instead. ◀

In the following, we assume that every instance satisfies the properties from Lemma 8.

▶ **Definition 9.** *We say that a solution for* IncMaxSep *is* represented *by a sequence of sizes* $(c_1, c_2, \dots)$ *if it first adds all elements from the set* $R_{c_1}$*, then all elements from the set* $R_{c_2}$*, and so on.*

A solution of IncMaxSep can only improve if it is altered in a way that it is represented by a sequence of sizes. Indeed, if not all elements of one set are added, the solution does not degrade if a smaller set is added instead because the density of the smaller set is at least as large as the density of the larger set. Moreover, adding all elements of one set consecutively is better because the value of the solution increases faster this way.

▶ **Lemma 10** ([1], Observation 2)**.** *There is an algorithm achieving the best-possible competitive ratio for* IncMaxSep *such that the solution generated by this algorithm can be represented by a sequence* $(c_1, c_2, \dots)$*. We can assume that* $v_{c_i} < v_{c_{i+1}}$ *and thus, since the values* $(v_i)_{i\in\mathbb{N}}$ *are non-decreasing,* $c_i < c_{i+1}$ *for all* $i \in \mathbb{N}$.

From now on, we will only consider solutions of this form and denote a solution $X$ by the sequence it is represented by, i.e., $X = (c_1, c_2, \dots)$. For a size $C \in \mathbb{N}$, we denote by $X(C)$ the first $C$ elements added by $X$, i.e., $|X(C)| = C$ and, with $O_i := \arg\max\{f(S) \mid S \subseteq U, |S| = i\}$, we have $X\left(\sum_{i=1}^{k} c_i\right) = \bigcup_{i=1}^{k} O_{c_i}$.

▶ **Proposition 11.** *The competitive ratios of* IncMax *and* IncMaxSep *coincide.*[1]

**Proof Sketch.** As IncMaxSep is a subclass of IncMax, the competitive ratio of IncMaxSep is not larger than that of IncMax.

It remains to show that the competitive ratio of IncMax is smaller or equal to that of IncMaxSep. To see this, consider an instance of IncMax. We will construct an instance of IncMaxSep such that every $\rho$-competitive solution to this problem instance induces a $\rho$-competitive solution for the initial instance of IncMax.

To define the instance of IncMaxSep, let $R_1, R_2, \dots$ be disjoint sets with $|R_i| = i$ for all $i \in \mathbb{N}$. For $i \in \mathbb{N}$, let $d_i = \textsc{Opt}(i)/i$. By modularity of the value function within one set $R_i$, the value of the optimal solution of a given size in this instance is the same as that in the instance of IncMax.

For $\rho \geq 1$, let $(c_1, c_2, \dots)$ be a $\rho$-competitive solution for the separable instance. We consider the solution for the initial problem that starts by adding the optimal solution of size $c_1$, then adds the optimal solution of size $c_2$, and so on. Accountability guarantees that it is possible to add the elements within one optimal solution such that the value of the partially added solution grows at least proportionally with the size of the solution. Since the values of the optimal solutions of a given size in the two instances coincide, the value of the solution for the initial instance we defined above is always greater or equal to that of the solution $(c_1, c_2, \dots)$. Thus, the solution for the initial instance is also $\rho$-competitive, which implies that the competitive ratio of IncMax is smaller or equal to that of IncMaxSep. ◀

## 3 Continuization Results

In order to find lower bounds on the competitive ratio of IncMaxSep, we transform the problem into a continuous one.

▶ **Definition 12.** *In the* IncMaxCont *problem, we are given a* density function $d\colon \mathbb{R}_{\geq 0} \to (0, 1]$ *and a* value function $v(c) := cd(c)$. *As for the discrete problem, we denote an incremental solution $X$ for* IncMaxCont *by a sequence of sizes $X = (c_1, c_2, \dots)$. For a given size $c \geq 0$, we denote the solution of this size by $X(c)$. With $n \in \mathbb{N}$ such that $\sum_{i=1}^{n-1} c_i < c \leq \sum_{i=1}^{n} c_i$, the value of $X(c)$ is defined as*

$$f(X(c)) := \max\left\{ \max_{i \in \{1,\dots,n-1\}} v(c_i), \left(c - \sum_{i=1}^{n-1} c_i\right) d(d_n) \right\}.$$

*An incremental solution $X$ is $\rho$-competitive if $\rho \cdot f(X(c)) \geq v(c)$ for all $c > 0$. The competitive ratio of $X$ is defined as* $\inf\{\rho \geq 1 \mid X$ *is $\rho$-competitive*$\}$.

The interpretation of the functions $d$ and $v$ is that the instance is partitioned into sets, one for every positive size $c \in \mathbb{R}$, each consisting of $c$ fractional units with a value of $d(c)$ per unit, for a total value of $v(c)$ for the set. The solution can be interpreted in the following way: It starts by adding the set of size $c_1$, then the set of size $c_2$, and so on. With $n \in \mathbb{N}$ such that

[1] A full proof of this and all other results can be found in [5].

$\sum_{i=1}^{n-1} c_i < c \leq \sum_{i=1}^{n} c_i$, the solution $X(c)$ has added all of the sets of sizes $c_1, \ldots, c_{n-1}$ and $c - \sum_{i=1}^{n-1} c_i$ units of the set of size $c_n$. Unlike the IncMaxSep problem, the IncMaxCont problem includes subsets of all real sizes instead of only integer sizes and, furthermore, allows fractional elements to be added to solutions instead of only an integral number of elements.

As for the discrete version of the problem, without loss of generality, we assume that the density function $d$ is non-increasing and the value function $v$ is non-decreasing. These assumptions imply that $d$ is continuous: If this was not the case and $d$ was not continuous for some size $c'$, i.e., $\lim_{c \nearrow c'} d(c) > \lim_{c \searrow c'} d(c)$, then $\lim_{c \nearrow c'} v(c) > \lim_{c \searrow c'} v(c)$ by definition of $v$, i.e., $v$ would not be increasing in $c$. So $d$ is continuous, and, by definition of $v$, also $v$ is continuous. Furthermore, without loss of generality, we assume that $d(0) = 1$.

For a fixed size $c \geq 0$, we define $p(c) = \max\{c' \geq 0 \mid v(c') \leq \rho v(c)\}$. This value gives the size up to which a solution with value $v(c)$ is $\rho$-competitive. Throughout our analysis, we assume that $p(c)$ is defined for every $c \geq 0$, i.e., that $\lim_{c \to \infty} v(c) = \infty$. Otherwise, any algorithm can terminate when the value of its solution is at least $\frac{1}{\rho} \sup_{c \in \mathbb{R}_{\geq 0}} v(c)$.

▶ **Proposition 13.** *The competitive ratio of IncMaxSep is greater or equal to that of IncMaxCont.*

**Proof Sketch.** Given a lower bound construction for the competitive ratio of IncMaxCont, one can discretize it with arbitrary resolution such that, with an arbitrarily small loss, it carries over to the IncMaxSep problem. ◀

This proposition implies that instead of devising a lower bound for the IncMaxSep problem, we can construct a lower bound for the IncMaxCont problem.

Note that it is not clear whether the competitive ratio of IncMaxSep and IncMaxCont coincide. This is due to the fact that a solution to the IncMaxCont problem may add fractional elements while a solution to the IncMaxSep problem may only add an integral number of items. There are even discrete instances where every continuization of the instance has a competitive ratio smaller than the initial instance.

▶ **Observation 14.** *There exists an instance of IncMaxSep that has a competitive ratio that is strictly larger than that of every instance of IncMaxCont that monotonically interpolates the IncMaxSep instance.*

**Proof Sketch.** We show that the instance of IncMaxSep with $N = 16$ sets and

$$\begin{aligned} d_1 &= 1, \\ d_3 = d_4 &= \frac{17}{40}, \\ d_{12} = d_{13} = d_{14} = d_{15} = d_{16} &= \frac{16473}{107200}. \end{aligned}$$

has a competitive ratio of at least 1.446, while every monotone interpolation of it has a competitive ratio of at most 1.425. ◀

Note that, even though this shows that there are instances where the continuous problem is easier than the discrete one, this does not rule out that the competitive ratios of IncMaxSep and IncMaxCont coincide. This is due to the fact that the instance in the proof is not a worst-case instance.

## 3.1 Optimal Continuous Online Algorithm

In this section, we present an algorithm to solve the IncMaxCont problem, and analyze it. To get an idea what the algorithm does, consider the following lemma. It gives a characterization of a solution $(c_1, c_2, \dots)$ being $\rho$-competitive, depending on $(c_1, c_2, \dots)$, $v$ and $d$.

▶ **Lemma 15.** *A solution $(c_1, c_2, \dots)$ for an instance of the IncMaxCont problem is $\rho$-competitive if and only if $d(c_1) \geq \frac{1}{\rho}$ and, for all $i \in \mathbb{N}$, $d(c_{i+1}) \geq \frac{v(c_i)}{p(c_i) - \sum_{j=1}^{i} c_j}$.*

The intuition behind the fraction

$$\frac{v(c_i)}{p(c_i) - \sum_{j=1}^{i} c_j}$$

is the following: The value of the solution $(c_1, \dots, c_{i-1}, c_i)$ is $v(c_i)$ and this value is $\rho$-competitive up to size $p(c_i)$. The size required for this solution is $\sum_{j=1}^{i} c_j$. Thus, in order to stay competitive, the size added next, namely $c_{i+1}$, needs to be chosen such that $\big(p(c_i) - \sum_{j=1}^{i} c_j\big) d(c_{i+1}) \geq v(c_i)$, i.e., the density $d(c_{i+1})$ is large enough such that the value of the solution of size $p(c_i)$ is $\big(p(c_i) - \sum_{j=1}^{i} c_j\big) d(c_{i+1})$.

We use this fraction to define an algorithm for solving the IncMaxCont Problem. For the algorithm, we assume that $v$ is strictly increasing and $d$ is strictly decreasing to make the choice of our algorithm unique. Every instance of IncMaxCont can be transformed to satisfy this with an arbitrarily small loss by simpliy "tilting" constant parts of $d$ and $v$ by a small amount. The algorithm GreedyScaling$(c_1, \rho)$ starts by adding the optimal solution of size $c_1 > 0$ and chooses the size $c_{i+1}$ such that

$$d(c_{i+1}) = \frac{v(c_i)}{p(c_i) - \sum_{j=1}^{i} c_j}, \tag{1}$$

i.e., as large as possible while still satisfying the inequality in Lemma 15. An illustration of the algorithm can be found in Figure 2.

Using the definition of the algorithm in (1) and Lemma 15, we are able to prove the following.

▶ **Proposition 16.** *The algorithm GreedyScaling$(c_1, \rho)$ is $\rho$-competitive if and only if it produces a solution $(c_1, c_2, \dots)$ with $c_i < c_{i+1}$ for all $i \in \mathbb{N}$ and $d(c_1) \geq \frac{1}{\rho}$.*

**Proof Sketch.** "$\Leftarrow$": If $c_i < c_{i+1}$ for all $i \in \mathbb{N}$ and $d(c_1) \geq 1/\rho$, we can simply apply Lemma 15 and obtain that the solution is $\rho$-competitive.

"$\Rightarrow$": If $d(c_1) < 1/\rho$, Lemma 15 yields that the solution is not $\rho$-competitive. If $c_{k+1} \leq c_k$ for some $k \in \mathbb{N}$, one can iteratively show that $c_{i+1} \leq c_i$ for all $i \in \{k, k+1, \dots\}$. This implies that the value of the solution $(c_1, c_2, \dots)$ is smaller or equal to $v(c_k)$ for all sizes. Yet, for large sizes $C \in \mathbb{N}$, we have $v(C) > \rho v(c_k)$ as $\lim_{c \to \infty} v(c) = \infty$. ◀

The algorithm GreedyScaling$(c_1, \rho)$ only depends on the desired competitive ratio $\rho$ and the starting value $c_1$. Given that some algorithm can achieve a competitive ratio of $\rho$, we can show that GreedyScaling$(c_1^*, \rho)$ with the correct starting value $c_1^* > 0$ also gives a $\rho$-competitive solution.

▶ **Theorem 1.** *For every instance of IncMaxCont, there exists a starting value $c_1$ such that the algorithm GreedyScaling$(c_1, \rho^*)$ achieves the best-possible competitive ratio $\rho^* \geq 1$.*

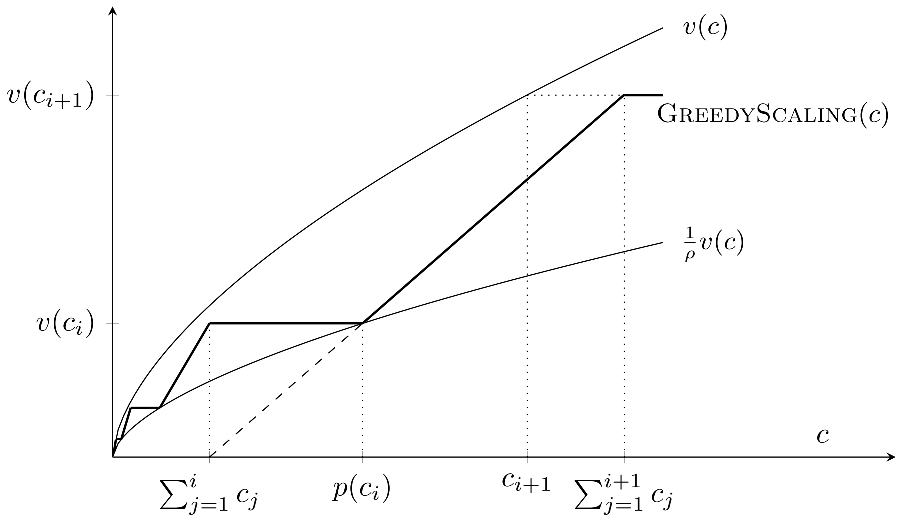

**Figure 2** Illustration how GreedyScaling$(c_1, \rho)$ works. Between size $\sum_{j=1}^{i} c_j$ and size $\sum_{j=1}^{i+1} c_j$, the algorithm adds the optimal solution of size $c_{i+1}$. This size is chosen in a way that the value of the partially added solution has value $v(c_i)$ exactly at size $p(c_i)$, i.e., when the previously added solution of size $c_i$ loses $\rho$-competitiveness.

**Proof sketch.** The idea of the proof is to start with a $\rho^*$-competitive solution $(c_1, c_2, \dots)$ for the instance of IncMaxCont. For every $k \in \mathbb{N}$, we define a new $\rho^*$-competitive solution $(c_1^k, c_2^k, \dots)$ as follows. For $i \in \mathbb{N}$ with $\sum_{j=1}^{i} c_j \geq k$, we set $c_i^k = c_i$. For $i \in \mathbb{N}$ with $\sum_{j=1}^{i} c_j < k$, we choose $c_i^k \geq 0$ as small as possible without losing $\rho^*$-competitiveness. This new solution satisfies the inequality

$$d(c_{i+1}^k) \geq \frac{v(c_i^k)}{p(c_i^k) - \sum_{j=1}^{i} c_j^k}$$

from Lemma 15 with equality for $i \in \{1, \dots, k-1\}$. This implies, that we can calculate $c_2^k, \dots, c_k^k$ solely based on $c_1^k$, $d$, and $v$. For every $k \in \mathbb{N}$, we obtain such a solution $(c_1^k, c_2^k, \dots)$. For all $k \in \mathbb{N}$, we have $d(c_1^k) \geq 1/\rho^*$, which implies that all sizes in $\{c_1^1, c_1^2, \dots\}$ are from the finite interval $[0, d^{-1}(1/\rho^*)]$. By the Bolzano-Weierstrass theorem, this implies that the sequence $(c_1^1, c_1^2, \dots)$ contains a converging sub-sequence. If we choose the limit of this sub-sequence to be the starting value $c_1^*$ of the algorithm GreedyScaling$(c_1^*, \rho^*)$, we obtain a $\rho^*$-competitive algorithm. ◀

For a range of starting values $c_1$, we are able to show the upper bound on the competitive ratio of GreedyScaling$(c_1, \varphi + 1)$ in Theorem 2, where $\varphi = \frac{1}{2}(1 + \sqrt{5}) \approx 1.618$ is the golden ratio.

▶ **Theorem 2.** *GreedyScaling$(c_1, \varphi + 1)$ is $(\varphi + 1)$-competitive if and only if $d(c_1) \geq \frac{1}{\varphi+1}$.*

**Proof Sketch.** By Proposition 16, it suffices to show that, for the solution $(c_1, c_2, \dots)$ produced by GreedyScaling$(c_1, \varphi + 1)$, we have $c_{i+1} > c_i$ for every $c_1 > 0$ with $d(c_1) > \frac{1}{\varphi+1}$. We show iteratively that $c_{i+1} \geq (\varphi + 1)c_i$. In order to do this, we observe that

$$p(c_i) = \frac{(\varphi + 1)v(c_i)}{d(p(c_i))} \geq (\varphi + 1)c_i.$$

By a straighforward induction that uses the fact that $(\varphi+1)^{i-j}c_j \leq c_i$ for all $j \in \{1,\dots,i-1\}$ as well as the definition of $\textsc{GreedyScaling}(c_1, \varphi+1)$ we obtain that $d(c_{i+1}) < d(p(c_i))$. This implies $c_{i+1} > p(c_i) \geq (\varphi+1)c_i$. ◀

Since $\textsc{GreedyScaling}(c_1, \rho)$ with the correct starting value $c_1$ is the best-possible algorithm for a fixed instance, we can give a lower bound of $\rho > 1$ for the $\textsc{IncMaxCont}$ problem by finding an instance that is a lower bound for $\textsc{GreedyScaling}(c_1, \rho)$ with all starting values $c_1 > 0$ that satisfy $d(c_1) \leq 1/\rho$. In the following, we show that, for every countable set of starting values, there is an instance where $\textsc{GreedyScaling}(c_1, \rho)$ cannot have a competitive ratio of better than $\varphi + 1$ for any of these starting values. In order to do this, we need the following lemma.

▶ **Lemma 17.** *For $\alpha, \beta, \rho, \epsilon \in \mathbb{R}_{\geq 0}$ with $\beta > 0$, consider the recursively defined sequence $(t_n)_{n\in\mathbb{N}}$ with*

$$t_0 = \beta, \qquad t_{n+1} = \frac{1}{\frac{\rho}{t_n(1-\epsilon)} - \left(\sum_{j=0}^{n} \frac{(\rho+\epsilon)^{j-n}}{t_j}\right) - \frac{\alpha}{(\rho+\epsilon)^n}} \quad \text{for all } n \in \mathbb{N} \cup \{0\}.$$

*If $1 < \rho < \varphi + 1$, then there exists $\epsilon' > 0$ such that, for all $\epsilon \in (0, \epsilon']$, there is $\ell \in \mathbb{N}$ with $t_\ell < 0$.*

**Proof sketch.** We define an auxiliary sequence $(a_n)_{n\in\mathbb{N}}$ with $a_n = \frac{1}{t_n}$ for all $n \in \mathbb{N} \cup \{0\}$. This sequence becomes negative if and only if $(t_n)_{n\in\mathbb{N}\cup\{0\}}$ becomes negative. We show that $(a_n)_{n\in\mathbb{N}\cup\{0\}}$ is fully described by the homogeneous recurrence relation

$$a_{n+1} = a_n\left(\frac{1}{\rho+\epsilon} + \frac{\rho}{1-\epsilon} - 1\right) - a_{n-1}\frac{\rho}{(1-\epsilon)(\rho+\epsilon)}$$

for all $n \in \mathbb{N}$, together with the start values $a_0 = 1/\beta$ and

$$a_1 = \frac{1}{t_1} = \frac{\rho}{\beta(1-\epsilon)} - \frac{1}{\beta} - \alpha.$$

Its characteristic polynomial is

$$0 = x^2 - \left(\frac{1}{\rho+\epsilon} + \frac{\rho}{1-\epsilon} - 1\right)x + \frac{\rho}{(1-\epsilon)(\rho+\epsilon)}.$$

We show that the roots $x$ and $y$ of this polynomial are complex if $\rho < \varphi + 1$ and $\epsilon > 0$ small enough. Thus, they are also distinct which implies that the sequence $(a_n)_{n\in\mathbb{N}\cup\{0\}}$ has the closed-form expression

$$a_n = \lambda x^n + \mu y^n$$

for all $n \in \mathbb{N} \cup \{0\}$ where $\lambda, \mu \in \mathbb{C}$ are chosen accordingly. The fact that the starting values $a_0$ and $a_1$ are real valued imply that $\lambda$ and $\mu$ are complex conjugate. Thus, we obtain

$$a_n = 2\Re(\lambda x^n)$$

for all $n \in \mathbb{N} \cup \{0\}$, where $\Re(\lambda x^n)$ denotes the real part of $\lambda x^n$. We analyze this equation by visualizing it on the complex plane (cf. Figure 3). Since $x$ is not real valued, multiplying by $x$ corresponds to a rotation by an angle that is not 0 and not $\pi$. Thus, for some $n \in \mathbb{N} \cup \{0\}$, $\Re(\lambda x^n)$ must become negative. ◀

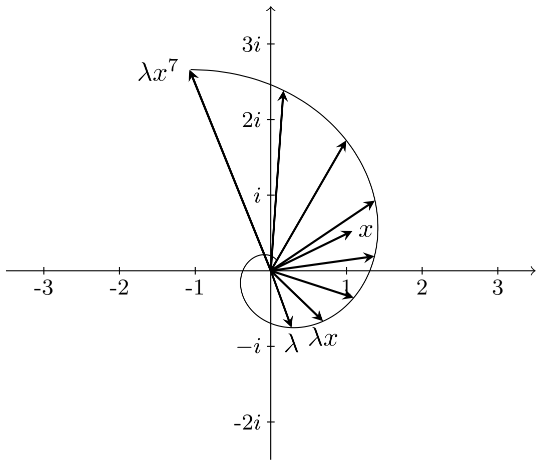

**Figure 3** Multiplying $\lambda$ repeatedly by $x \in (\mathbb{C} \setminus \mathbb{R})$ is equivalent to a rotation around the origin that, at some point, reaches the half-plane corresponding to negative real parts.

▶ **Proposition 3.** *For every countable set $S \subset \mathbb{R}_{>0}$ of starting values, there exists an instance of* IncMaxCont *such that* GreedyScaling$(c_1, \rho)$ *is not $\rho$-competitive for any $c_1 \in S$ and any $\rho < \varphi + 1$.*

**Proof Sketch.** We give an overview how to construct an instance where the algorithm GreedyScaling$(c_1, \rho)$ is not $\rho$-competitive for one fixed starting value $c_1 > 0$ and every $\rho \in [1, \varphi + 1)$. For the sake of simplicity, in this overview, we describe an instance where the density function $d$ and the value function $v$ are locally constant. In the final construction, we avoid this by slightly tilting constant parts of the function.

Let $\epsilon > 0$ be arbitrarily small. The beginning of the instance up to size $c_1$ can be chosen arbitrarily. We set $d(c) = d(c_1)$ for all $c \in [c_1, (\rho + \epsilon)c_1]$. By doing this, we ensure that the value obtained by adding the optimal solution of the first size $c_1$ is $\rho$-competitive for as few sizes as possible, i.e., until $p(c_1) = \rho c_1$. Then, $d(c_2) = \frac{v(c_1)}{\rho c_1 - c_1}$ can be calculated. We set $v(c) = v((\rho+\epsilon)c_1)$ for all $c \in [(\rho + \epsilon)c_1, \frac{v((\rho+\epsilon)c_1)}{d(c_2)}]$. This ensures that $c_2$ is as small as possible, namely $c_2 = \frac{v((\rho+\epsilon)c_1)}{d(c_2)}$. Now we repeat what we did for $c_1$, i.e., we define $d$ to be constant so that the value $v(c_2)$ is $\rho$-competitive for as few sizes as possible. Then, we calculate $d(c_3)$ and define $v$ to be constant so that $c_3$ is as small as possible. We continue doing this for all larger $c_i$ with $i \geq 3$. It turns out that we have $d(c_i) = t_i$ where the sequence $(t_i)_{i \in \mathbb{N}}$ is defined as in Lemma 17. Thus, at some point, the density GreedyScaling$(c_1, \rho)$ calculates the next capacity to be negative, which is not possible, i.e., the algorithm is not $\rho$-competitive.

We have seen how to construct an instance that excludes one starting value. This instance is finite and the beginning can be chosen arbitrarily. Thus, we can chain together multiple of these instances by scaling an instance for some set of starting values and modifying the beginning such that it contains an instance for an additional starting value. ◀

## 3.2 General Lower Bound

Now we want to employ the techniques we used to prove Lemma 17 and Proposition 3 in order to prove a lower bound on the competitive ratio of IncMaxCont. Let $\rho^*$ be the unique real root $\rho \geq 1$ of the polynomial $-4\rho^6 + 24\rho^4 - \rho^3 - 30\rho^2 + 31\rho - 4$. As before, we need to show that a recursively defined sequence becomes negative at some point.

▶ **Lemma 18.** *For $\rho \in \mathbb{R}_{\geq 0}$ and $\epsilon > 0$, consider the recursively defined sequence $(t_n)_{n\in\mathbb{N}}$ with*

$$t_0 = 1, \qquad t_1 = \frac{1-\epsilon}{\rho}, \qquad t_n = \frac{1-\epsilon}{\frac{\rho}{t_{n-1}} - \frac{1}{t_{n-2}} - \frac{1}{\rho}\left(\sum_{j=0}^{n-3} \frac{(\rho+\epsilon)^{j+2-n}}{t_j}\right)} \quad \text{for all } n \in \mathbb{N}_{\geq 2}.$$

*If $1 < \rho < \rho^*$, then there exists $\epsilon' > 0$ such that, for all $\epsilon \in [0, \epsilon']$, there is $\ell \in \mathbb{N}$ with $t_\ell < 0$.*

The proof of this lemma is along the same lines as the proof of Lemma 17, with additional technical difficulties because the recurrence relation of the sequence is of order 3. With this lemma, we are ready to construct our lower bound on the competitive ratio of IncMaxCont and thus, via Propositions 11 and 13, of IncMax.

▶ **Theorem 4.** *The IncMax problem has a competitive ratio of at least 2.246.*

**Proof sketch.** We fix a competitive ratio $\rho < \rho^*$ and some small $\epsilon > 0$. Similarly to the construction in the proof of Proposition 3, the lower bound in Theorem 4 is a construction where we have intervals on which, alternatingly, either the density function or the value function is constant (cf. Figure 4). For $i \in \mathbb{N}$, on the $(2i)$-th interval, the value is constant and equals $(\rho+\epsilon)^{i-1}$. On the $(2i-1)$-th interval, the density is constant and equal to $t_{i-1}$, where $(t_n)_{n\in\mathbb{N}}$ is defined as in Lemma 18. Every solution that contains a size from an interval of constant value can be improved by picking the largest size from the preceding interval of constant density instead. This size has the same value and is smaller. Thus, we assume that algorithms only pick sizes from the intervals with constant density. We denote the solution by $(c_1, c_2, \dots)$. We have $d(c_1) = t_0 = 1$ because $t_1 < 1/\rho$ is too small. In order to be competitive for the first constant value interval of value 1, the solution has to satisfy $c_1 \geq 1/\rho$ to achieve a value of at least $1/\rho$. Then, the following recursive argument is made. Fix $i \in \mathbb{N}$. Whenever, for all $j \in \{1, \dots, i\}$, the solution satisfies $d(c_j) = t_{j-1}$ and $c_j \geq \frac{(\rho+\epsilon)^{i-1}}{\rho}$, then we have $d(c_{i+1}) = t_i$ and $c_{i+1} \geq \frac{(\rho+\epsilon)^i}{\rho}$. The equality $d(c_{i+1}) = t_i$ is due to the definition of the sequence $(t_n)_{n\in\mathbb{N}}$ and Lemma 15. The inequality $c_{i+1} \geq \frac{(\rho+\epsilon)^i}{t_i \rho}$ follows from the fact that, after the size $c_{i+1}$ is added to the solution, the solution has to be competitive on the $(2i+2)$-th interval of value $(\rho+\epsilon)^i$. Since the sequence $(t_n)_{n\in\mathbb{N}}$ becomes negative at some point, the solution is not $\rho$-competitive. ◀

## 4 Randomized Incremental Maximization

We turn to analyzing randomized algorithms to solve the (discrete) IncMaxSep problem. In contrast to deterministic algorithms, we do not compare the value obtained by the algorithm to an optimum solution, but rather the expected value obtained by the algorithm. This enables us to find an algorithm with randomized competitive ratio smaller than the lower bound of 2.24 on the competitive ratio of deterministic algorithms in Theorem 4.

### 4.1 Randomized Algorithm

Scaling algorithms, i.e., algorithms where the size $c_i$ is chosen such that $c_i = \delta c_{i-1}$ with an appropriate scaling factor $\delta > 1$, have been proven to perform well for the deterministic version of the problem. The best known algorithm is, in fact, a scaling algorithm [1]. In the analysis, it turns out that, on average, a scaling algorithm performs better than the actual competitive ratio, which is only tight for few sizes. By randomizing the initial size $c_0$, we manage to average out the worst-case sizes in the analysis.

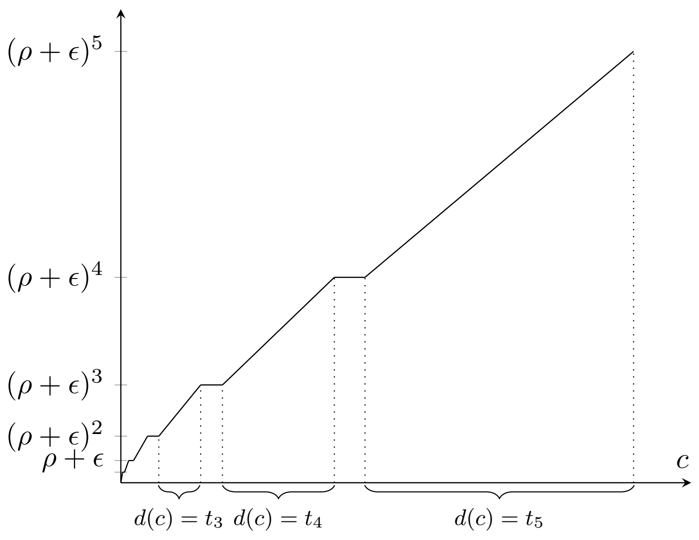

**Figure 4** Lower bound construction for $\rho = 2.1$.

We describe the randomized algorithm RandomizedScaling for IncMaxSep. Let $r > 1$ be some scaling parameter to be determined later. The algorithm RandomizedScaling starts by choosing $\epsilon \in (0,1)$ uniformly at random. For all $i \in \mathbb{N}_0$, it calculates $\tilde{c}_i := r^{i+\epsilon}$ and $c_i := \lfloor \tilde{c}_i \rfloor$ and returns the solution $(c_0, c_1, c_2, \dots)$.[2] This approach is similar to a randomized algorithm to solve the CowPath problem in [16], which also calculates such a sequence with a different choice of $r \in \mathbb{R}$ in order to explore a star graph.

We define

$$\tilde{t}_i := \sum_{j=0}^{i} \tilde{c}_j = r^\epsilon \frac{r^{i+1}-1}{r-1} \qquad \text{and} \qquad t_i := \sum_{j=0}^{i} c_j.$$

For better readability, we let $\tilde{c}_{-1} = c_{-1} = \tilde{t}_{-1} = t_{-1} = 0$. Note that, for all $i \in \mathbb{N}_0$, we have

$$t_{i-1} \le \tilde{t}_{i-1} = r^\epsilon \frac{r^i - 1}{r-1} \overset{r \ge 2}{\le} r^{i+\epsilon} - r^\epsilon \le r^{i+\epsilon} - 1 = \tilde{c}_i - 1 \le c_i. \tag{2}$$

For every size $c \in \mathbb{N}_0$, we denote the solution created by the algorithm RandomizedScaling by $X_{\text{ALG}}(c)$. Note that the optimum solution of size $c \in \mathbb{N}_0$ is given by the set $R_c$ because $v_1 \le v_2 \le \dots$ and $d_1 \ge d_2 \ge \dots$. Thus, the value of the optimum solution of size $c$ is $v_c$.

In order to find an upper bound on the randomized competitive ratio of Randomized-Scaling, we need the following lemma. It gives an estimate on the expected value of the solution for a fixed size $C \in \mathbb{N}$ of RandomizedScaling depending on the interval in which $C$ falls.

▶ **Lemma 19.** *Let $C \in \mathbb{N}$.*

1. *For $i \in \mathbb{N} \cup \{0\}$ with $\mathbb{P}[C \in (c_{i-1}, c_i]] > 0$, we have*

$$\mathbb{E}\big[f(X_{\text{ALG}}(C)) \mid C \in (c_{i-1}, c_i]\big] \ge \mathbb{E}\Big[\max\Big\{\frac{c_{i-1}}{C}, \frac{C - t_{i-1}}{\max\{C, c_i\}}\Big\} \,\Big|\, C \in (c_{i-1}, c_i]\Big] \cdot v_C.$$

[2] With this definition, the algorithm does not terminate on finite instances. To avoid this, it suffices to stop calculating the sizes $c_i$ until they are larger than the number of elements in the instance.

2. *For $i \in \mathbb{N}$ with $\mathbb{P}[C \in (\tilde{c}_i, \tilde{t}_i - 1]] > 0$, we have*

$$\mathbb{E}\big[f(X_{\text{ALG}}(C)) \mid C \in (\tilde{c}_i, \tilde{t}_i - 1]\big] \geq \mathbb{E}\Big[1 - \frac{\tilde{t}_{i-1}}{C} \;\Big|\; C \in (\tilde{c}_i, \tilde{t}_i - 1]\Big] \cdot v_C.$$

3. *For $i \in \mathbb{N}$ with $\mathbb{P}[C \in (\tilde{t}_{i-1} - 1, \tilde{c}_i]] > 0$, we have*

$$\mathbb{E}\big[f(X_{\text{ALG}}(C)) \mid C \in (\tilde{t}_{i-1} - 1, \tilde{c}_i]\big] \geq \mathbb{E}\Big[\max\Big\{\frac{\tilde{c}_{i-1} - 1}{C}, \frac{C - \tilde{t}_{i-1}}{\tilde{c}_i}\Big\} \;\Big|\; C \in (\tilde{t}_{i-1} - 1, \tilde{c}_i]\Big] \cdot v_C.$$

By choosing $r \approx 5.1646$ to be the unique maximum of

$$\begin{aligned} g(x) \quad = \quad & \frac{1 - \sqrt{\left(\frac{x^3-1}{x-1}x^z - 1\right)^2 + 4x^{5+2z}}}{2\log(x)x^{3+z}} - (1-\delta)\frac{1 - x^{-3}}{x-1} + z - \frac{1 - x^{-3}}{2(x-1)\log(x)} \\ & - \Big(\frac{1 - x^{-3}}{x-1} - \frac{1}{x^{3+z}}\Big)\Big(\log_x\Big(\sqrt{\Big(\frac{x^3-1}{x-1}x^z - 1\Big)^2 + 4x^{5+2z}} - \frac{x^3-1}{x-1}x^z + 1\Big) \\ & - \log_x(2) - 3\Big) - \frac{2x^{2+z}}{\Big(\sqrt{\left(\frac{x^3-1}{x-1}x^z - 1\right)^2 + 4x^{5+2z}} - \frac{x^3-1}{x-1}x^z + 1\Big)\log(x)} \\ & + \frac{2}{\log(x)} - \Big(1 + \frac{1}{x^{3+z}}\Big)\Big(\log_x(x^{3+z} + 1) + \log_x(x-1) - \log_x(x^4 - 1)\Big), \end{aligned}$$

we can show that the following holds.

▶ **Lemma 20.** *Let $k \in \mathbb{N}$ and $\delta \in (0, 1]$ such that $r^{k+\delta} \geq \sum_{i=0}^{3} r^i$. Then*

$$\begin{aligned} g(r) \leq I(k, \delta) := & \int_{\min\{1,\mu(k-1)\}}^{1} 1 - \frac{\tilde{t}_{k-2}}{r^{k+\delta}}\, d\epsilon \qquad + \int_{\min\{1,\nu(k-1)\}}^{\min\{1,\mu(k-1)\}} \frac{\tilde{c}_{k-1} - 1}{r^{k+\delta}}\, d\epsilon \\ & + \int_{\delta}^{\min\{1,\nu(k-1)\}} \frac{r^{k+\delta} - \tilde{t}_{k-1}}{\tilde{c}_k}\, d\epsilon + \int_{\max\{0,\mu(k)\}}^{\delta} 1 - \frac{\tilde{t}_{k-1}}{r^{k+\delta}}\, d\epsilon \\ & + \int_{\max\{0,\nu(k)\}}^{\max\{0,\mu(k)\}} \frac{\tilde{c}_k - 1}{r^{k+\delta}}\, d\epsilon \qquad + \int_{0}^{\max\{0,\nu(k)\}} \frac{r^{k+\delta} - \tilde{t}_k}{\tilde{c}_{k+1}}\, d\epsilon \end{aligned}$$

*where*

$$\begin{aligned} \mu(i) = & \log_r(r^{k+\delta} + 1) + \log_r(r - 1) - \log_r(r^{i+1} - 1), \\ \nu(i) = & \log_r\Big(\sqrt{\Big(r^{k+\delta}\frac{1 - r^{-(i+1)}}{r-1} - 1\Big)^2 + 4r^{2k+2\delta-1}} - r^{k+\delta}\frac{1 - r^{-(i+1)}}{r-1} + 1\Big) - \log_r(2) - i. \end{aligned}$$

With these lemmas, we are ready to prove an upper bound of $1/g(r) < 1.772$ on the randomized competitive ratio of RANDOMIZEDSCALING.

▶ **Theorem 5.** *INCMAX admits a 1.772-competitive randomized algorithm.*

**Proof Sketch.** In order to find this estimate, we start by fixing $k \in \mathbb{N}$ such that $C \in [r^k, r^{(k+1)})$. Then, depending on the value of $\epsilon$, $C$ is from one of the intervals

$$I_1 = (\tilde{c}_{k-1}, \tilde{t}_{k-1} - 1], \quad I_2 = (\tilde{t}_{k-1} - 1, \tilde{c}_k], \quad I_3 = (\tilde{c}_k, \tilde{t}_k - 1], \quad I_4 = (\tilde{t}_k - 1, \tilde{c}_{k+1}].$$

Yet, not all of these intervals are relevant to calculate the randomized competitive ratio. Depending on where in the interval $[r^k, r^{(k+1)})$ the value $C$ lies, only 2 or 3 of the intervals $I_1$ to $I_4$ have a non-zero probability to contain $C$. Thus, we distinguish the different cases, where $C$ lies in $[r^k, r^{(k+1)})$ and use Lemma 19 to calculate the randomized competitive ratio to be the integral expression in Lemma 20. Applying this lemma gives the desired bound on the randomized competitive ratio. ◀

## 4.2 Randomized Lower Bound

We turn to proving the lower bound in Theorem 6 for IncMaxSep.

▶ **Theorem 6.** *Every randomized* IncMax *algorithm has competitive ratio at least* 1.447.

**Proof.** We fix $N$ to be the number of sets $R_1, \dots, R_N$, leaving $d_1, \dots, d_N$ as parameters to determine the instance; we denote the resulting instance by $I(d_1, \dots, d_N)$. Note that, given a probability distribution $p_1, \dots, p_N$ over the elements $\{1, \dots, N\}$ in addition, Yao's principle [24] yields

$$\inf_{\textsc{Alg} \in \mathcal{A}_N} \sum_{i=1}^{N} p_i \cdot \frac{i \cdot d_i}{\textsc{Alg}(I(d_1, \dots, d_N), i)}$$

as a lower bound on the randomized competitive ratio of the problem. Here, $\textsc{Alg}(I, i)$ denotes the value of the first $i$ elements in the solution produced by Alg on instance $I$, and $\mathcal{A}_N$ is the set of all deterministic algorithms on instances with $N$ sets $R_1, \dots, R_N$. As observed earlier, we may assume that

$$\mathcal{A}_N := \left\{ \textsc{Alg}_{c_1, \dots, c_\ell} \;\middle|\; 1 \le c_1 < \dots < c_\ell \le N, \sum_{i=1}^{\ell} c_i \le N \right\},$$

where $\textsc{Alg}_{c_1, \dots, c_\ell}$ is the algorithm that first includes all elements of $R_{c_1}$ into the solution, then all elements of $R_{c_2}$, and so on. Once it has added the $c_\ell$ elements of $R_{c_\ell}$, it adds some arbitrary elements from then onwards.

We can formulate the problem of maximizing the lower bound on the competitive ratio as an optimization problem:

$$\begin{aligned} \max \quad & \rho \\ \text{s.t.} \quad & \rho \le \sum_{i=1}^{N} p_i \cdot \frac{i \cdot d_i}{\textsc{Alg}(I(d_1, \dots, d_N), i)} & \forall \textsc{Alg} \in \mathcal{A}_N, \\ & \sum_{i=1}^{N} p_i = 1, \\ & d_1, \dots, d_N \ge 0, \\ & p_1, \dots, p_N \ge 0. \end{aligned}$$

Note that the expression $\textsc{Alg}_{c_1, \dots, c_\ell}(I(d_1, \dots, d_N), i)$ can also be written as a function of $c_1, \dots, c_\ell$, $d_1, \dots, d_N$, and $i$ by taking the maximum over all sets from which $\textsc{Alg}_{c_1, \dots, c_\ell}$ selects elements:

$$\textsc{Alg}_{c_1, \dots, c_\ell}(I(d_1, \dots, d_N), i) = \max_{1 \le j \le \ell} \left\{ \max \left\{ i - \sum_{1 \le j' < j} c_{j'}, c_j \right\} \cdot d_{c_j} \right\}.$$

A feasible solution to the above optimization problem with $N = 10$ is given by

$$\begin{aligned} &(\rho; d_1, \dots, d_{10}; p_1, \dots, p_{10}) \\ &\quad = (1.447; 1, 1/2, 1/2, 1/2, 2/5, 1/3, 1/3, 1/3, 1/3, 1/3; 0.132, 0, 0, 0.395, 0, 0, 0, 0, 0, 0.473), \end{aligned}$$

with objective value 1.447. ◀

## References

1 Aaron Bernstein, Yann Disser, Martin Groß, and Sandra Himburg. General bounds for incremental maximization. *Math. Program.*, 191(2):953–979, 2022. doi:10.1007/s10107-020-01576-0.

2 Avrim Blum, Prasad Chalasani, Don Coppersmith, William R. Pulleyblank, Prabhakar Raghavan, and Madhu Sudan. The minimum latency problem. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing (STOC)*, pages 163–171. ACM, 1994. doi:10.1145/195058.195125.

3 Marek Chrobak, Claire Kenyon, John Noga, and Neal E. Young. Incremental medians via online bidding. *Algorithmica*, 50(4):455–478, 2008. doi:10.1007/s00453-007-9005-x.

4 Yann Disser, Max Klimm, Nicole Megow, and Sebastian Stiller. Packing a knapsack of unknown capacity. *SIAM J. Discret. Math.*, 31(3):1477–1497, 2017. doi:10.1137/16M1070049.

5 Yann Disser, Max Klimm, Kevin Schewior, and David Weckbecker. Incremental maximization via continuization. arXiv:2305.01310v1.

6 Yann Disser, Max Klimm, and David Weckbecker. Fractionally subadditive maximization under an incremental knapsack constraint. In *Proceedings of the 19th International Workshop on Approximation and Online Algorithms (WAOA)*, pages 206–223. Springer, 2021. doi:10.1007/978-3-030-92702-8_13.

7 Ryo Fujita, Yusuke Kobayashi, and Kazuhisa Makino. Robust matchings and matroid intersections. *SIAM J. Discret. Math.*, 27(3):1234–1256, 2013. doi:10.1137/100808800.

8 Michel X. Goemans and Jon M. Kleinberg. An improved approximation ratio for the minimum latency problem. *Math. Program.*, 82:111–124, 1998. doi:10.1007/BF01585867.

9 Michel X. Goemans and Francisco Unda. Approximating incremental combinatorial optimization problems. In *Proceedings of the 20th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 6:1–6:14, 2017. doi:10.4230/LIPIcs.APPROX-RANDOM.2017.6.

10 Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985. doi:10.1016/0304-3975(85)90224-5.

11 Jeff Hartline and Alexa Sharp. An incremental model for combinatorial maximization problems. In *Proceedings of the 5th International Workshop on Experimental Algorithms (WEA)*, pages 36–48, 2006. doi:10.1007/11764298_4.

12 Jeff Hartline and Alexa Sharp. Incremental flow. *Networks*, 50(1):77–85, 2007. doi:10.1002/net.20168.

13 Refael Hassin and Shlomi Rubinstein. Robust matchings. *SIAM J. Discret. Math.*, 15(4):530–537, 2002. doi:10.1137/S0895480198332156.

14 Refael Hassin and Danny Segev. Robust subgraphs for trees and paths. *ACM Trans. Algorithms*, 2(2):263–281, 2006. doi:10.1145/1150334.1150341.

15 Naonori Kakimura and Kazuhisa Makino. Robust independence systems. *SIAM J. Discret. Math.*, 27(3):1257–1273, 2013. doi:10.1137/120899480.

16 Ming-Yang Kao, John H Reif, and Stephen R Tate. Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. *Information and Computation*, 131(1):63–79, 1996.

17 Max Klimm and Martin Knaack. Maximizing a submodular function with bounded curvature under an unknown knapsack constraint. In *Proceedings of the 25th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 49:1–49:19, 2022. doi:10.4230/LIPIcs.APPROX/RANDOM.2022.49.

18 Guolong Lin, Chandrashekhar Nagarajan, Rajmohan Rajaraman, and David P. Williamson. A general approach for incremental approximation and hierarchical clustering. *SIAM J. Comput.*, 39(8):3633–3669, 2010. doi:10.1137/070698257.

19 Jannik Matuschke, Martin Skutella, and José A. Soto. Robust randomized matchings. *Math. Oper. Res.*, 43(2):675–692, 2018. doi:10.1287/moor.2017.0878.

20 Nicole Megow and Julián Mestre. Instance-sensitive robustness guarantees for sequencing with unknown packing and covering constraints. In *Proceedings of the 4th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 495–504, 2013. doi:10.1145/2422436.2422490.

21 Julián Mestre. Greedy in approximation algorithms. In *Proceedings of the 14th Annual European Symposium on Algorithms (ESA)*, pages 528–539, 2006. doi:10.1007/11841036_48.

22 Ramgopal R. Mettu and C. Greg Plaxton. The online median problem. *SIAM J. Comput.*, 32(3):816–832, 2003. doi:10.1137/S0097539701383443.

23 C. Greg Plaxton. Approximation algorithms for hierarchical location problems. *J. Comput. Syst. Sci.*, 72(3):425–443, 2006. doi:10.1016/j.jcss.2005.09.004.

24 Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 222–227, 1977. doi:10.1109/SFCS.1977.24.