Online Algorithms for Set Packing with Renewable Capacities

Anya Chaturvedi
School of Computing and Augmented Intelligence
Arizona State University, Tempe, AZ, USA

William K. Moses Jr.
Department of Computer Science
Durham University, UK

Christian Scheideler
Department of Computer Science
Paderborn University, Paderborn, Germany

Prudence W.H. Wong
School of Computer Science and Informatics
University of Liverpool, UK

Abstract

We propose and study a new extension of the classical set packing problem, which we call the online D-set packing with renewable capacities (D-SPaRC) problem. In the D-SPaRC setting, there is a collection of resources with associated capacities. Requests arrive one by one, and for each request, a decision has to be made to accept it or not before seeing future requests. Each request is associated with a collection of subsets of resources, with each subset having cardinality at most D. When accepting a request, exactly one of these subsets must be chosen, which consumes one unit of capacity on each of the involved resources. Over time, the available resource capacities may be renewed, allowing additional requests to be accepted. Many online problems, including packing, routing, and scheduling, can be formulated as a D-SPaRC problem, thus underlining its usefulness.

We first present a simple greedy algorithm that is $O(\hat{c}_{\min} \cdot (D^{1/\hat{c}_{\min}} - 1))$-competitive for $D \geq 2$ and 3-competitive if $D = 1$, where $\hat{c}_{\min}$ is the minimum capacity of the resources. We then show that, for $\hat{c}_{\min} = \Omega(\log D)$, the greedy algorithm can be extended to an online algorithm with predictions whose competitive ratio is never worse than that of the original greedy approach and can, in fact, be reduced to a constant when the predictions are optimal. Finally, we generalize the D-SPaRC problem in two meaningful ways, namely by addressing non-uniform request priorities and by handling requests with non-uniform weights.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Set packing, online algorithms, learning-augmented algorithms

Digital Object Identifier 10.4230/LIPIcs.SAND.2026.10

Funding *Christian Scheideler*: DFG Projects SCHE 1592/10-1 and 1592/11-1.

1 Introduction

The *online D-set packing with renewable capacities (D-SPaRC) problem* consists of a set V of resources (simply called *nodes* in the following) with associated capacities $\hat{c} : V \to \mathbb{N}$. Let $\hat{c}_{\min} = \min_v \hat{c}(v)$. Initially, the *available* capacity $c(v)$ at every node v is equal to $\hat{c}(v)$. Requests $r_1, r_2, \ldots$ arrive one by one, where each r_i is a non-empty collection of non-empty subsets of V, with each subset having cardinality at most D (representing the maximum allowed resource demand per request). For each r_i, a decision must be made to accept or reject it before seeing future requests. A request r_i can only be accepted if there exists a set $S \in r_i$ with $c(v) \geq 1$ for all $v \in S$. We call such sets *feasible*. If r_i is accepted, exactly one

feasible set S must be picked, and $c(v)$ is decreased by 1 for all nodes $v \in S$. Independent of the decisions and unknown to the online algorithm, the node capacities are renewed over time, i.e., there is a sequence of capacity renewal vectors $c_1, c_2, \ldots$ over V, where $c_i(v) \geq 0$ specifies the amount of the available capacity at node v that is renewed between requests r_i and r_{i+1}. More specifically, given a renewal of $c_i(v)$, the available capacity of v is updated to $\min\{\hat{c}(v), c(v) + c_i(v)\}$ before request r_{i+1} is proposed. The primary objective of this paper is to accept as many requests as possible, and the quality of our solutions is measured using the *competitive ratio*, which is the worst-case ratio of the performance of an optimal offline algorithm to that of the proposed online algorithm over any sequence of requests. We will also discuss some extensions at the end of this paper.

The D-SPaRC problem has interesting applications in the context of renewable energy. Consider, for example, a future scenario in which windmills recharge batteries for package-delivering drones. Given a request to send a package from a source A to a destination B, where B is too far away from A to deliver the package on a single battery charge, several optional flight paths—in which drone batteries can be replaced en route at windmills—can then be used to ensure the package is delivered to B. Any collection of such windmills may be represented as a set, and therefore, a request to deliver a package can be modeled as a collection of subsets of the set of windmills. This can be formulated as a D-SPaRC problem if the number of times a battery is allowed to be replaced for a single package is upper-bounded by D. We illustrate this using an example in Figure 1. Instead of windmills at fixed locations, one can also envision a scenario in which batteries are available on mobile units, like trucks, so that certain drone routes are only temporarily available.

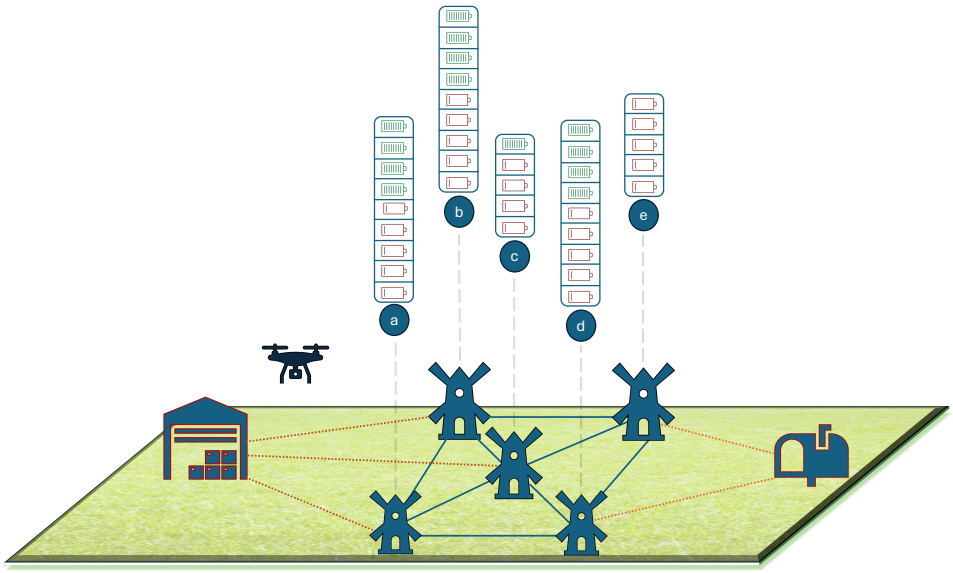

Figure 1 Illustration of the D-SPaRC problem applied to drone-based package delivery, where a drone must transport a package from a warehouse (source, left) to a mailbox (destination, right). Windmills serve as intermediate battery-replacement stations, represented as nodes, where each windmill v has maximum capacity $\hat{c}(v)$, depicted as a vertical stack of batteries, where green and red batteries indicate available and depleted capacity, respectively. An edge between two nodes indicates the drone can travel between them on a single battery charge. For this instance, consider the request $r_i = \{\{a, c, d\}, \{a, b, e\}\}$ with $D = 3$. r_i can be accepted using the feasible subset $\{a, c, d\}$ (while the other is infeasible since e has no remaining capacity).

In general, we may consider any situation in which there is a collection of suppliers of a certain type of renewable resource, and a request is only allowed to use those combinations of suppliers that do not exceed the allowed budget of any given request while satisfying its needs. In that case, we may end up with weighted combinations of suppliers, which can be modeled as multisets and can be covered by one of our extensions. The D-SPaRC formulation can also cover scenarios in which suppliers are temporarily out of service (i.e., unable to deliver their available resources), provided this occurs independently of the online algorithm's choices. In that case, a request contains only multisets of suppliers currently in service.

1.1 Our Contributions

We present a new variant of the set packing problem and show that a simple greedy algorithm is $O(\hat{c}_{\min} \cdot (D^{1/\hat{c}_{\min}} - 1))$-competitive for $D \geq 2$ and 3-competitive if $D = 1$, where $\hat{c}_{\min}$ is the minimum capacity of the resources. Its proof is inspired by an analysis of a greedy algorithm for the unsplittable flow problem [23], but the new aspect here is that we are dealing with renewable resources, which significantly complicates the witness argument in the proof.

We then show that, for $\hat{c}_{\min} = \Omega(\log D)$, the greedy algorithm can be extended to an online algorithm with predictions whose competitive ratio is never worse than that of the original greedy approach and can, in fact, be reduced to a constant when the predictions are optimal. The key idea here is to compare the performance of an optimal algorithm with the performance of the greedy algorithm and an optimal algorithm that only have half of the capacity available in the resources, and whenever the capacity of a resource is renewed by an additive c in the former case, it is only renewed by $c/2$ in the latter case. It turns out that the greedy algorithm still remains $O(\hat{c}_{\min} \cdot (D^{2/\hat{c}_{\min}} - 1))$-competitive in that case, and the optimal algorithm for half of the capacities still remains constant competitive if $\hat{c}_{\min} = \Omega(\log D)$. Combining these two insights then allows us to design an online algorithm with predictions that is $O(\log D)$-competitive for any predictions and constant competitive if the predictions are optimal, under the assumption that $\hat{c}_{\min} = \Omega(\log D)$.

Finally, we generalize the D-SPaRC problem in two meaningful ways, namely by addressing non-uniform request priorities and by handling requests with non-uniform weights. In fact, given a maximum priority of $p_{\max}$, a competitive ratio of $O(\hat{c}_{\min} \cdot (D^{\log(p_{\max}+1)/\hat{c}_{\min}} - 1))$ can be reached, and given a maximum weight of $\hat{\mu}$, a competitive ratio of $O(\hat{c}_{\min} \cdot (D^{1/(\hat{c}_{\min}-\hat{\mu})} - 1))$ can be reached.

1.2 Related Work

Set packing is a fundamental problem that belongs to Karp's 21 $\mathcal{NP}$-complete problems [22]. Suppose that we are given a finite set S and a collection $\mathcal{C}$ of subsets of S. Then, the (decision variant of the) set packing problem asks if there are k pairwise disjoint subsets in $\mathcal{C}$. The set packing problem is not only $\mathcal{NP}$-complete, but its optimization version has been shown to be as hard to approximate as the maximum clique problem by Hazan, Safra, and Schwartz [19], and the best known algorithm approximates it within a factor of $O(\sqrt{|S|})$ (see the work of Halldórsson, Kratochvíl, and Telle [18]). A more tractable problem is the *D-set packing problem* in which each set contains at most D elements. When $D = 1$, the problem is trivial, and when $D = 2$, the problem is equivalent to finding a maximum cardinality matching, which can be solved in polynomial time. For any $D \geq 3$, the problem is $\mathcal{NP}$-hard, as it is a generalization of the D-dimensional matching problem, and Karp [22] showed that 3-dimensional matching is $\mathcal{NP}$-hard. However, there are constant-factor approximation

algorithms (e.g., Cygan [9], Fürer and Yu [15]).

D-set packing is equivalent to D-hypergraph matching where the sets of size at most D correspond to hyperedges of size at most D. *Online D-hypergraph matching* has been studied by Tröbst and Udwani [30], where they show that any (randomized) online algorithm has competitive ratio at least $(2 + o(1))/D$. (Note that they define competitive ratio inversely to the definition used in this paper, i.e., they consider the ratio of the algorithm performance to the optimal performance.) They also study the version of the problem where edges are allowed to be assigned fractionally. Notice that D-hypergraph matching differs from D-SPaRC in two ways: the hyperedges (sets) are disjoint and the nodes are not renewable.

A closely related problem to our formulation of the online D-set packing problem (without considering renewable capacities) is the *online unsplittable flow problem (UFP)*, also referred to as the online unsplittable routing problem. In online UFP, we are given an undirected graph consisting of n nodes and positive real-valued edge capacities. Each request comes in as a tuple of its associated source and destination nodes, along with a corresponding demand $d \in [0, 1]$. The goal is to maximize the number of requests accepted. We can successfully satisfy a request if there exists a path in the graph such that each involved edge has sufficient capacity to satisfy the given demand d (and the flow is never split amongst multiple paths). Once a request is accepted, the capacities along the chosen path will be updated accordingly. One can observe that online UFP, where each demand is exactly 1, can be modeled using our formulation of online D-set packing, where each element is an edge, and each collection of elements represents a potential flow path to satisfy a given request. Furthermore, D would be an upper bound on the length of any flow path in the graph. The online version of the UFP was first studied by Awerbuch, Azar, and Plotkin [2], who considered the case where the minimum edge capacity is $\Omega(\log n)$ and designed an $O(\log n)$-competitive algorithm. Over the years, various works have continued to study the problem, aiming to improve the competitive ratio by leveraging different parameters (e.g., flow number by Kolman and Scheideler [23]) or different techniques (e.g., primal-dual technique by Buchbinder and Naor [8]).

Motivated by packet forwarding, Emek, Halldórsson, Mansour, Patt-Shamir, Radhakrishnan, and Rawitz [12, 13] introduced a formulation of the online set packing problem where initially we are aware of several subsets of S whose elements are unknown, and elements arrive in an online manner, revealing which subset(s) they belong to. The goal is to irrevocably assign incoming elements to one of their potential subsets such that the total number of subsets with all elements assigned is maximized. For this formulation, they developed a randomized distributed algorithm to solve the problem and showed a nearly matching lower bound on the competitive ratio that holds even in the centralized setting. This version of the online set packing problem, where subsets are inputs and elements are requests, is a variant of D-SPaRC where the roles are reversed. Thus, solutions to D-SPaRC are not applicable to this formulation.

There is also some overlap between D-SPaRC and the *online knapsack problem*. In the online knapsack problem, a knapsack of some capacity receives elements one by one, each with a specific weight and value. An irrevocable decision to accept or reject the element is made, and the goal is to maximize the total value of the elements accepted to the knapsack without exceeding its capacity. One may notice that for the version of online knapsack where all elements have the same value but possibly different integer-value weights, this may be modeled as the non-uniform weight version of the D-SPaRC problem (see Section 4). The work of Marchetti-Spaccamela and Vercellis [29] was the first to study the online knapsack problem and showed that in the general case, there is no online algorithm that can achieve a non-trivial competitive ratio. The work of Zhou, Chakrabarty, and Lukose [32] subsequently

studied this problem by assuming that the weight of each item is much smaller than the knapsack capacity and that the value-weight ratio of each item is lower and upper bounded by two constants L and U, respectively. They were able to design an algorithm that solves the online knapsack with competitive ratio $\ln(U/L) + 1$, which they showed to be optimal. There has been much subsequent research on the problem, and the interested reader may refer to the survey by Böckenhauer, Hromkovič, Komm, Rossmanith, and Stocker [5].

Another line of work related to D-SPaRC is augmenting online algorithms with replenishment, as done by Kang, Liu, and Udwani [21]. They introduce a black-box method that extends many existing online resource allocation algorithms (for resources with fixed capacities) to work with an arbitrary (adversarial or stochastic) replenishment process.

As mentioned earlier, *drone routing* can be modeled via the D-SPaRC problem. In particular, consider a graph where the vertices are windmills at which drone batteries can be stored/charged. Each node's capacity represents the number of drone batteries that can be stored/charged at that node. The input is a sequence of delivery requests, where each request consists of a sequence of at most D nodes to be traversed by a drone (to deliver a package from a starting point to an ending point). The goal is to satisfy the maximum number of requests, where a request is satisfied if each node in the assignment has non-zero capacity when the request is being serviced. Drone routing is a more recent variant of the vehicle routing problem, introduced by Dantzig and Ramser [11], a source of active research for the past 67 years. An excellent survey on various work that has been done in this area is that of Laporte [25]. Macrina, Di Puglia Pugliese, Guerriero, and Laporte [28] provide a wonderful survey of work focused on drone routing.

Many works have considered providing additional information to an online algorithm in order to improve its performance. One important model is the advice model, where an oracle has advance knowledge of the entire input, chooses to encode some information in a string of bits, and makes this string available to the algorithm throughout the runtime to aid in the computation. For more information on this advice model, please see the relevant section in Komm [24] and the survey by Boyar, Favrholdt, Kudahl, Larsen, and Mikkelsen [6]. Typically, the oracle is assumed to be correct. *Learning-augmented algorithms*, also known as *algorithms with predictions*, attempt to capture more realistic situations where the information, potentially generated by a machine learning algorithm, may not always be correct. In this model, introduced by Lykouris and Vassilvitskii [27], the prediction may or may not be correct. Algorithms are then designed to leverage correct predictions to achieve better performance while withstanding incorrect predictions, i.e., performing as well as if no prediction was available. To the best of our knowledge, there has been no prior attempt to study the set packing problem, let alone the D-SPaRC problem, in a learning-augmented model. However, in this setting, the knapsack problem has been studied [1, 3, 7, 10, 16, 17, 20, 26, 31].

It should be noted that there are many packing problems that may seem related to set packing, such as bin packing, but are, in fact, different and the solutions to these problems do not immediately apply to set packing.

2 Bounded Greedy Algorithm

For any resource $v \in V$ and $x \in \{0, \ldots, \hat{c}(v)\}$ (specifying the unavailable capacity in v) let $f_v(x) = D^{x/\hat{c}(v)}$. Let the *weight* of a set $S \subseteq V$ be defined as $f(S) = \sum_{v \in S} f_v(\hat{c}(v) - c(v))$. Our *Bounded Greedy Algorithm (BGA)* works as follows: Let W be a suitable parameter. Given a request r, accept it whenever there is a feasible set $S \in r$ with $f(S) \leq W$, and pick

any such set in this case. Otherwise, reject it. We show the following result.

▶ **Theorem 1.** *The BGA with $W = 2D$ is $O(\hat{c}_{\min} \cdot (D^{1/\hat{c}_{\min}} - 1))$-competitive for $D \geq 2$ and 3-competitive for $D = 1$.*

Proof. First, consider the case $D \geq 2$. Let $\mathcal{B}$ be the set of requests accepted by the BGA and $\mathcal{O}$ be the set of requests accepted by the optimal offline solution OPT. Furthermore, let $\mathcal{O}' = \mathcal{O} \setminus \mathcal{B}$. Our goal will be to show that $|\mathcal{O}'|$ can be upper-bounded in terms of $|\mathcal{B}|$, which will then allow us to determine the competitive ratio of the BGA.

Consider any request $r \in \mathcal{O}'$. Since r was not accepted by the BGA, it must hold for the set $S(r)$ taken by OPT that

1. $S(r)$ was infeasible or
2. $f(S(r)) > W$

at the point when the BGA considered r. We will capture these two cases via some appropriate potential function. For any $v \in V$ let the potential of v be defined as $\phi(v) = (\hat{c}_{\min}/\hat{c}(v)) \sum_{i=0}^{\hat{c}(v)-c(v)-1} f_v(i)$, where $c(v)$ is the available capacity at v when the BGA considered r, and let $\phi(S(r)) = \sum_{v \in S(r)} \phi(v)$. It holds that

$$\begin{aligned}\phi(v) &= \frac{\hat{c}_{\min}}{\hat{c}(v)} \sum_{i=0}^{\hat{c}(v)-c(v)-1} D^{i/\hat{c}(v)} = \frac{\hat{c}_{\min}}{\hat{c}(v)} \cdot \frac{D^{(\hat{c}(v)-c(v))/\hat{c}(v)} - 1}{D^{1/\hat{c}(v)} - 1} \\ &= \frac{\hat{c}_{\min}}{\hat{c}(v)(D^{1/\hat{c}(v)} - 1)} \cdot (f_v(\hat{c}(v) - c(v)) - 1)\end{aligned}$$

Since $\hat{c}(v)(D^{1/\hat{c}(v)} - 1)$ is monotonically decreasing with increasing $\hat{c}(v)$ and therefore attains its maximum at $\hat{c}_{\min}(D^{1/\hat{c}_{\min}} - 1)$,

$$\phi(v) \geq \frac{1}{D^{1/\hat{c}_{\min}} - 1} \cdot (f_v(\hat{c}(v) - c(v)) - 1)$$

In case 1, there must be a node $v \in S(r)$ with $c(v) = 0$ and, therefore,

$$\phi(S(r)) \geq \phi(v) \geq \frac{D-1}{D^{1/\hat{c}_{\min}} - 1}$$

In case 2, we see that

$$\begin{aligned}\phi(S(r)) &= \sum_{v \in S(r)} \phi(v) \geq \frac{1}{D^{1/\hat{c}_{\min}} - 1} \sum_{v \in S(r)} (f_v(\hat{c}(v) - c(v)) - 1) \\ &\geq \frac{1}{D^{1/\hat{c}_{\min}} - 1} \cdot (f(S(r)) - D) \geq \frac{D-1}{D^{1/\hat{c}_{\min}} - 1}\end{aligned}$$

because in case 2, $f(S(r)) > W = 2D$.

Thus, in both cases, $\phi(S(r)) \geq \frac{D-1}{D^{1/\hat{c}_{\min}}-1}$. Therefore,

$$\begin{aligned}|\mathcal{O}'| &= \sum_{r \in \mathcal{O}'} 1 = \frac{D^{1/\hat{c}_{\min}} - 1}{D-1} \sum_{r \in \mathcal{O}'} \frac{D-1}{D^{1/\hat{c}_{\min}} - 1} \\ &\leq \frac{D^{1/\hat{c}_{\min}} - 1}{D-1} \sum_{r \in \mathcal{O}'} \phi(S(r))\end{aligned}$$

Next, we will upper bound the right-hand side in terms of $|\mathcal{B}|$. For that, we have to determine how to formulate suitable weights for the requests in $\mathcal{B}$ so that they cover $\sum_{r \in \mathcal{O}'} \phi(S(r))$.

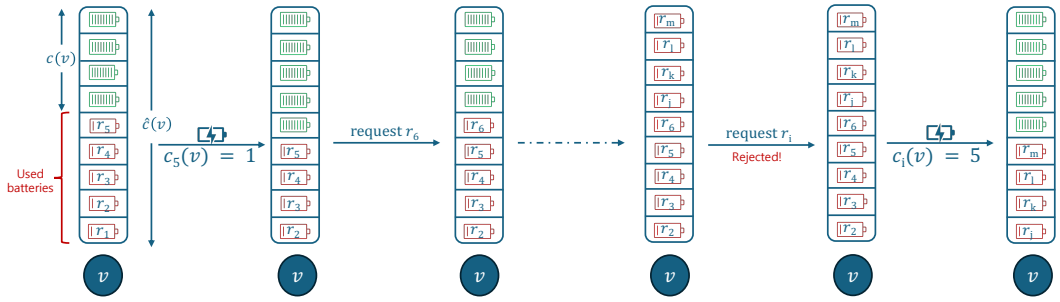

Figure 2 Illustration of the algorithm's progression and the charging argument at a single node v. Initially, $c(v) = \hat{c}(v)$. Requests $r_1, r_2, \ldots$ arrive one by one; each accepted request r_i is assigned the battery at height $\hat{c}(v) - c(v)$ (i.e., the lowest available slot), and $c(v)$ is decremented by 1 (used batteries are labeled by the request that consumed them). A request is rejected if no available capacity remains at v (shown for r_i, marked *Rejected!*). Between consecutive requests, capacity renewals may replenish $c(v)$ up to $\hat{c}(v)$; upon renewal, batteries associated with earlier requests are replenished first, shifting the remaining used batteries down accordingly.

Note that if $c(v) = \hat{c}(v)$, i.e., v has not been used by any request or its capacity has been fully recharged in the meantime, $\phi(v) = 0$. Thus, whenever $\phi(v) > 0$, this is witnessed by requests whose BGA-sets include v. Given a request $r \in \mathcal{O}'$ and a node $v \in S(r)$, let $c(v)$ be the available capacity at v when the BGA considers r. Consider the capacity $\hat{c}(v)$ of v to be represented by a stack of $\hat{c}(v)$ many slots with heights 0 to $\hat{c}(v) - 1$, where the $\hat{c}(v) - c(v)$ lowest slots are occupied by the BGA-requests that were last accepted by the BGA and whose sets include v, ordered from the oldest at slot 0 to the youngest at slot $\hat{c}(v) - c(v) - 1$ (see Figure 2). If we now charge a weight of $(\hat{c}_{\min}/\hat{c}(v))f_v(i)$ to each request of height i, we can cover $\phi(v)$ in $\phi(S(r))$. To determine the maximum total weight a request $r' \in \mathcal{B}$ can contribute to node v in $\sum_{r \in \mathcal{O}'} \phi(S(r))$, we first observe that the initial height of r' at v is $\hat{c}(v) - c_{r'}(v)$, where $c_{r'}(v)$ is the available capacity at v when considering r', and afterwards the height of r' in v monotonically decreases over time. More precisely, whenever the capacity at v gets renewed by γ, the height of r' drops by γ, and once the height of r' drops below 0 this way, it will not be considered for future requests accepted by OPT. Certainly, r' cannot be charged for an arbitrary number of requests in OPT because once v has been used by $\hat{c}(v)$ many requests accepted by OPT, OPT can only choose v for additional requests if its capacity gets renewed. Hence, the total weight that needs to be charged to r' w.r.t. node v gets maximized if it stays at its initial height for $\hat{c}(v)$ many OPT requests and additionally $\hat{c}(v) - c_{r'}(v)$ many OPT requests at height h for all $h \in \{0, \ldots, \hat{c}(v) - c_{r'}(v) - 1\}$. Thus, the maximum total weight that a request $r' \in \mathcal{B}$ contributes to node v in $\sum_{r \in \mathcal{O}'} \phi(S(r))$ is at most

$$\hat{c}(v) \cdot \frac{\hat{c}_{\min}}{\hat{c}(v)} f_v(\hat{c}(v) - c_{r'}(v)) + \frac{\hat{c}_{\min}}{\hat{c}(v)} \cdot \sum_{i=0}^{\hat{c}(v)-c_{r'}(v)-1} f_v(i)$$
$$\leq \left(\hat{c}_{\min} + \frac{\hat{c}_{\min}}{\hat{c}(v)(D^{1/\hat{c}(v)} - 1)}\right) f_v(\hat{c}(v) - c_{r'}(v))$$

From the equation $e^x = \sum_{i \geq 0} x^i/i!$ it easily follows that $\lim_{x \to \infty} x(D^{1/x} - 1) = \ln D$ and we already stated above that the function $x(D^{1/x} - 1)$ is monotonically decreasing for $x \geq 1$. Thus, $\hat{c}(v)(D^{1/\hat{c}(v)} - 1) \geq \ln D$ for all $D \geq 2$ and $\hat{c}(v) \geq 1$. Moreover, if $\hat{c}_{\min} = \epsilon \ln D$ then

$\hat{c}_{\min}(D^{1/\hat{c}_{\min}} - 1) = \epsilon \ln D(e^{1/\epsilon} - 1) \leq \epsilon(e^{1/\epsilon} - 1)\hat{c}(v)(D^{1/\hat{c}(v)} - 1)$. Thus, for $\hat{c}_{\min} = \epsilon \ln D$,

$$\begin{aligned}
&\left(\hat{c}_{\min} + \frac{\hat{c}_{\min}}{\hat{c}(v)(D^{1/\hat{c}(v)} - 1)}\right) f_v(\hat{c}(v) - c_{r'}(v)) \\
&\leq \left(\hat{c}_{\min} + \frac{\epsilon(e^{1/\epsilon} - 1)}{D^{1/\hat{c}_{\min}} - 1}\right) f_v(\hat{c}(v) - c_{r'}(v)) \\
&= \left(\hat{c}_{\min} + \frac{\hat{c}_{\min}}{\ln D}\right) f(\hat{c}(v) - c_{r'}(v)) = \hat{c}_{\min}\left(1 + \frac{1}{\ln D}\right) f_v(\hat{c}(v) - c_{r'}(v))
\end{aligned}$$

Therefore,

$$\begin{aligned}
|\mathcal{O}'| &\leq \frac{D^{1/\hat{c}_{\min}} - 1}{D - 1} \sum_{r \in \mathcal{O}'} \phi(S(r)) \leq \frac{D^{1/\hat{c}_{\min}} - 1}{D - 1} \sum_{r' \in \mathcal{B}} \hat{c}_{\min}\left(1 + \frac{1}{\ln D}\right) f(S(r')) \\
&= \hat{c}_{\min}\left(1 + \frac{1}{\ln D}\right) \frac{D^{1/\hat{c}_{\min}} - 1}{D - 1} \sum_{r' \in \mathcal{B}} f(S(r')) \\
&\leq 3\hat{c}_{\min} \frac{D^{1/\hat{c}_{\min}} - 1}{D - 1} \cdot W \cdot |\mathcal{B}| \\
&= O(\hat{c}_{\min}(D^{1/\hat{c}_{\min}} - 1)|\mathcal{B}|)
\end{aligned}$$

for $W = 2D$ and $D \geq 2$. Since $|\mathcal{O} \setminus \mathcal{O}'| \leq |\mathcal{B}|$, the theorem follows.

It remains to consider the case $D = 1$. For $D = 1$, $f_v(x)$ is always 1. Therefore, the BGA with $W = 2D = 2$ accepts any request r that has a set $\{v\} \in r$ with $c(v) > 0$, and picks any such node for r in this case. Thus, for any request $r \in \mathcal{O}'$, the node in $S(r)$ taken by r was infeasible for the BGA because $c(v) = 0$. This implies that

$$\phi(S(r)) = \frac{\hat{c}_{\min}}{\hat{c}(v)} \sum_{i=0}^{\hat{c}(v)-c(v)-1} f_v(i) = \hat{c}_{\min}$$

and therefore,

$$|\mathcal{O}'| \leq \frac{1}{\hat{c}_{\min}} \sum_{r \in \mathcal{O}'} \phi(S(r))$$

To cover $\sum_{r \in \mathcal{O}'} \phi(S(r))$, it suffices for every request r' accepted by the BGA to assign a weight of at most

$$\hat{c}(v) \cdot \frac{\hat{c}_{\min}}{\hat{c}(v)} + \frac{\hat{c}_{\min}}{\hat{c}(v)} \cdot (\hat{c}(v) - c_{r'}(v)) \leq 2\hat{c}_{\min}$$

to (the node v picked by) r'. Hence,

$$|\mathcal{O}'| \leq \frac{1}{\hat{c}_{\min}} \sum_{r' \in \mathcal{B}} 2\hat{c}_{\min} = 2|\mathcal{B}|$$

which implies that the BGA is 3-competitive in this case. ◀

Due to a lower bound of $\Omega(\hat{c}_{\min} \cdot D^{1/(\hat{c}_{\min}-1)})$ in [23] that holds for any $2 \leq \hat{c}_{\min} \leq \log D$ and any deterministic online algorithm for (a special case of) the D-SPaRC problem, the upper bound is almost optimal. The theorem also implies that for $D \geq 2$ and $\hat{c}_{\min} \geq \log D$, the BGA is $O(\log D)$-competitive.

3 Bounded Greedy Algorithm with Predictions

Next, we will investigate how the BGA can be combined with predictions so that, for the case that the predictions are good, a constant competitive ratio can be reached, while for the case that the predictions are bad, the BGA with predictions will have a competitive ratio that is not worse than the competitive ratio of the original BGA algorithm.

Consider any $\beta \in (0,1)$ with $\hat{c}_{\min} \geq 1/\beta$. First, we want to compare the performance of BGA(β) with OPT, where the bounded greedy algorithm BGA(β) can only use a β-fraction of the capacities of the nodes. More precisely, initially, for every node $v \in V$, the available capacity at v for BGA(β) is just $\beta \cdot \hat{c}(v)$, and it takes $1/\beta$ capacity renewals at v until the capacity for BGA(β) renews by 1. In this case, it holds:

▶ **Theorem 2.** *For any $\beta \in (0,1)$ with $\beta^{-1} \in \mathbb{N}$ and $\beta \cdot \hat{c}_{\min} \geq 1$, BGA($\beta$) with $W = 2D$ is $O(\hat{c}_{\min} \cdot (D^{1/(\beta \cdot \hat{c}_{\min})} - 1))$-competitive for $D \geq 2$ and $1 + 2/\beta$-competitive if $D = 1$.*

Proof. For simplicity, redefine the node capacities so that BGA(β) is operating on the original capacities $\hat{c}(v)$ while OPT is operating on the capacities $\beta^{-1}\hat{c}(v)$. Then the proof follows along the same lines as the proof of Theorem 1. The only difference is that we have to adapt the weight associated with a request $r' \in \mathcal{B}$ to cover $\sum_{r \in \mathcal{O}'} \phi(S(r))$ (where ϕ is defined as before because we consider here the situation of the BGA). More precisely, a request $r' \in \mathcal{B}$ might be used up to $\beta^{-1} \cdot \hat{c}(v)$ times as witness for some request accepted by OPT before changing its height due to capacity renewals, and afterwards, in the worst case up to β^{-1} times as witness for each height $h \in \{0, \ldots, \hat{c}(v) - c_{r'}(v) - 1\}$. This increases the weight associated with r' by a factor of β^{-1}, which increases the competitive ratio by a factor of β^{-1}, resulting in $O(\beta^{-1} \cdot \hat{c}_{\min} \cdot (D^{1/\hat{c}_{\min}} - 1))$. Substituting $\hat{c}(v)$ by $\beta \cdot \hat{c}(v)$ results in the bound in Theorem 2 for $D \geq 2$. The special case $D = 1$ follows along the same lines. ◀

Since the number of requests accepted by OPT is at least as large as the number of requests accepted by the BGA, Theorem 2 immediately implies the following result, where, analogously to BGA(1/2), OPT(1/2) is the optimal algorithm that has only half of the capacity of OPT.

▶ **Corollary 3.** *OPT(1/2) is $O(\hat{c}_{\min} \cdot (D^{2/\hat{c}_{\min}} - 1))$-competitive compared to OPT.*

For $\hat{c}_{\min} = \Omega(\log D)$, we can prove a stronger result.

▶ **Theorem 4.** *If $\hat{c}_{\min} \geq \gamma \log D$ for a sufficiently large constant γ, then OPT(1/2) is constant competitive compared to OPT.*

Proof. The proof uses the Lovász Local Lemma (LLL) [14]:

▶ **Lemma 5** (Lovász Local Lemma)**.** *Let $A_1, \ldots, A_n$ be events in an arbitrary probability space. Suppose that $G = (V, E)$ is a dependency graph of these events and that there are real numbers $x_i \in (0,1)$ for all $1 \leq i \leq n$ with*

$$\Pr[A_i] \leq x_i \prod_{\{i,j\} \in E} (1 - x_j)$$

for all $1 \leq i \leq n$. Then,

$$\Pr[\bigcap_{i=1}^{n} \bar{A}_i] \geq \prod_{i=1}^{n} (1 - x_i) \, .$$

In particular, with positive probability no bad event A_i holds.

Let $r_1, \ldots, r_n$ be the requests accepted by OPT for the original capacities, and let $S(r_i)$ be the set selected for r_i. We assume for the rest of the proof that $D \geq 2$ and n is above a sufficiently large constant since otherwise the theorem is trivially true. Consider the random experiment that for each of the accepted requests r_i, OPT(1/2) decides independently of the other requests with probability $p = (1-\epsilon)/2$ for some constant $0 < \epsilon < 1$ whether to accept r_i as well. Let event B be true iff OPT(1/2) accepts at most $p \cdot n/2$ requests. Then it follows from the Chernoff bounds that

$$\Pr[B] \leq e^{-(1/2)^2 p \cdot n/2} = e^{-(1-\epsilon)n/16}$$

Furthermore, for every node v and request r_i with $v \in S(r_i)$, let event $A(v,i)$ be true iff OPT(1/2) uses more than $\hat{c}(v)/2$ of v's capacity after considering r_i, i.e., the available capacity is less than $\hat{c}(v)/2$. To bound the probability that $A(v,i)$ is true, let $X(v,i)$ be the total amount of used capacity in v after OPT considered r_i and the random variable $X'(v,i)$ be the total amount of used capacity in v after OPT(1/2) considered r_i. Certainly, $\mathbb{E}[X'(v,i)] = p \cdot X(v,i) \leq p \cdot \hat{c}(v)$. Since the decision to accept a request accepted by OPT is done independently of the other requests, it follows from the Chernoff bounds that

$$\Pr[X'(v,i) \geq \hat{c}(v)/2] = \Pr[X'(v,i) \geq (1+(1-2p)/(2p))p \cdot \hat{c}(v)] \leq e^{-\frac{1-2p}{2p} p \cdot \hat{c}(v)/3} = e^{-\epsilon \hat{c}(v)/6}$$

To determine the dependencies between the events $A(v,i)$, we use w.l.o.g. the rule that whenever the used capacity of a node v is c, this is due to the last c requests r accepted by OPT with $v \in S(r)$. In this case, two events $A(v,i)$ and $A(w,j)$ are dependent if and only if $S(r_i) \cap S(r_j) \neq \emptyset$ and there is a node $v \in S(r_i) \cap S(r_j)$ where r_i and r_j witnessed the used capacity in v at the same time. Since r_i can only witness a used capacity in nodes $v \in S(r_i)$ and for each such v, r_i can witness a used capacity jointly with at most $2\hat{c}(v)$ many other requests, it follows that $A(v,i)$ can depend on at most $\sum_{w \in S(r_i)} 2\hat{c}(w)$ many other events $A(w,j)$. Furthermore, all events $A(v,i)$ depend on B. For all events $A(v,i)$ let us choose $x(v,i) = x(v) = \frac{\epsilon}{\hat{c}(v) \cdot D}$, and for event B choose $x(B) = e^{-(1-\epsilon)n/32}$. Then it follows that

$$\begin{aligned} x(v,i) \cdot (1-x(B)) \prod_{w \in S(r_i)} (1-x(w))^{2\hat{c}(w)} &\geq \frac{\epsilon}{\hat{c}(v) \cdot D} \cdot (1-x(B)) \prod_{w \in S(r_i)} e^{-\frac{2\epsilon}{\hat{c}(w) \cdot D} \cdot 2\hat{c}(w)} \\ &\geq \frac{\epsilon}{\hat{c}(v) \cdot D} \cdot (1-x(B)) \cdot e^{-4\epsilon} \geq e^{-\epsilon \hat{c}(v)/6} \end{aligned}$$

if $\hat{c}_{\min} \geq \gamma \log D$ for some sufficiently large constant γ and n is above a sufficiently large constant. Thus,

$$\Pr[X'(v,i) \geq \hat{c}(v)/2] \leq x(v,i) \cdot (1-x(B)) \prod_{w \in S(r_i)} (1-x(w))^{2\hat{c}(w)}$$

Moreover,

$$\begin{aligned} x(B) \cdot \prod_{i=1}^{n} \prod_{v \in S(r_i)} (1-x(v,i)) &= x(B) \cdot \prod_{i=1}^{n} \prod_{v \in S(r_i)} \left(1 - \frac{\epsilon}{\hat{c}(v) \cdot D}\right) \\ &\geq x(B) \cdot \prod_{i=1}^{n} \prod_{v \in S(r_i)} e^{-\frac{2\epsilon}{\hat{c}(v) \cdot D}} \\ &\geq x(B) \cdot \prod_{i=1}^{n} e^{-\frac{2\epsilon}{\hat{c}_{\min}}} = x(B) \cdot e^{-2\epsilon n/\hat{c}_{\min}} \end{aligned}$$

Since $e^{-2\epsilon n/\hat{c}_{\min}} \geq e^{-(1-\epsilon)n/32}$ if the constant γ in $\hat{c}_{\min}$ is large enough, it follows that

$$\Pr[B] \leq x(B) \cdot \prod_{i=1}^{n} \prod_{v \in S(r_i)} (1 - x(v,i))$$

Thus, due to the LLL, with positive probability, no bad event B or $A(v,i)$ holds, which means we have a valid solution for OPT(1/2) with constant competitiveness. ◀

The condition in Theorem 4 that $\hat{c}_{\min} \geq \gamma \log D$ is best possible up to constant factors, as shown in the following lemma.

▶ **Lemma 6.** *For $\hat{c}_{\min} = o(\log D)$, OPT(1/2) cannot be constant competitive compared to OPT.*

Proof. The *independence number* of a hypergraph $H = (V, E)$, denoted by $\alpha(H)$, is the maximum size of a subset $U \subseteq V$ that does not contain an edge of the hypergraph. Consider a random D-regular k-uniform hypergraph $H = (V, E)$ with $V = \{1, \ldots, n\}$. For any node i let $E(i)$ be the set of hyperedges containing i. To obtain an instance of the D-SPaRC problem, consider the requests $r_1, r_2, \ldots, r_n$ on the resource set E with $r_i = \{E(i)\}$ for all i. If all resources have capacity k, then all requests can be accepted because each hyperedge (representing a resource) is requested by exactly k requests. However, if all resources have a capacity of $k-1$, the maximum number of requests that can be accepted is equal to the independence number $\alpha(H)$. Bennett and Frieze [4] have shown that for random D-regular k-uniform hypergraphs H, $\alpha(H)$ converges to $n(\frac{k \log D}{(k-1)D})^{1/(k-1)}$ for any fixed k and D being sufficiently large, which implies that a constant factor difference between k and $k-1$ w.r.t. the maximum number of accepted requests is only achieved if $k = \Omega(\log D)$, resulting in the lemma. In fact, for the special case of $k = 2$, the difference is as high as $\Theta((\log D)/D)$, and therefore, for $\hat{c}_{\min} = 2$, there are instances of the D-SPaRC problem where OPT(1/2) is just $\Theta(D/(\log D))$-competitive compared to OPT. ◀

Now, we are ready to explain how to extend the BGA with predictions. Suppose, for simplicity, that all nodes have even capacities. In this case, we can evenly distribute the capacity of v among two copies v_1 and v_2 so that each copy has a capacity of $\hat{c}(v)/2$. Whenever there is a renewal of the capacity of v, the renewed capacity will alternately be given to v_1 and v_2. The available capacities in the copies v_1 are exclusively available for some oracle Ω that uses predictions, while the available capacities of the copies v_2 are exclusively available for the BGA. Whenever the BGA accepts a request, Ω is not consulted, but whenever the BGA does not accept a request r and there is at least one feasible set $S(r)$ w.r.t. the copies v_2, we consult Ω whether to accept the request, and if so, which feasible set $S(r)$ to take.

Let us call this BGA with predictions BGA(Ω). Combining Theorems 2 and 4, we get:

▶ **Theorem 7.** *If Ω is ω-competitive w.r.t. OPT(1/2) and $\hat{c}_{\min} \geq \gamma \log D$ for a sufficiently large constant γ then BGA(Ω) is $O(\min\{\omega, \log D\})$-competitive.*

Hence, our BGA with predictions can make the best use of the predictions without sacrificing the original competitive ratio of the original BGA.

4 Extensions

In this section, we study two extensions of the basic D-SPaRC problem.

4.1 Non-uniform priorities

Suppose that each request r_i has a priority $p(r_i) \in \mathbb{N}$ and the goal is to maximize the sum of the priorities of the accepted requests. Furthermore, suppose that the maximum priority $p_{\max}$ is known in advance to the online algorithm and that $\hat{c}_{\min} \geq \lceil \log(p_{\max} + 1) \rceil$. Then we can use the following *priority-based BGA algorithm*: Use the standard trick of cutting the requests into $k = \lceil \log(p_{\max} + 1) \rceil$ priority classes, where priority class $i \geq 0$ contains all priorities in the interval $[2^i, 2^{i+1} - 1)$, and reserve a $1/k$ fraction of the capacity of each node for each priority class. Moreover, capacity renewals are evenly distributed among the reserved capacities. Then it follows from Theorem 2:

▶ **Theorem 8.** *If $k = \lceil \log(p_{\max} + 1) \rceil$ and $\hat{c}_{\min} \geq k$ is an integer multiple of k, the priority-based BGA with $W = 2D$ is $O(\hat{c}_{\min} \cdot (D^{k/\hat{c}_{\min}} - 1))$-competitive for $D \geq 2$ and $O(k)$-competitive for $D = 1$.*

Hence, if $D \geq 2$ and $\hat{c}_{\min} \geq k \cdot \log D$ then the priority-based BGA is $O(k \cdot \log D)$-competitive.

4.2 Non-uniform weights

Suppose that we allow the sets a request can choose from to be arbitrary non-empty multisets of V of size at most D, where a node in V can occur up to μ times in a multiset S, for some $\mu > 1$. Note that in this case, $D \geq \mu > 1$ as well, i.e., we do not have to consider the special case $D = 1$. Then we can use the following *weighted BGA algorithm*: Given a request r, accept it whenever there is a feasible multiset $S \in r$ with $f(S) \leq W$, and pick any such set in this case. A multiset S is *feasible* if for every $v \in V$, $c(v) \geq \mu(v)$, where $\mu(v)$ is the number of times v appears in S, and the weight $f(S)$ is defined as

$$f(S) = \sum_{v \in S} \sum_{i=0}^{\mu(v)-1} f_v(\hat{c}(v) - c(v) + i)$$

where $f_v(x) = D^{x/\hat{c}(v)}$ is defined as before. Then it follows:

▶ **Theorem 9.** *If $\hat{c}_{\min} \geq \mu$ then the weighted BGA with $W = D + D^{1+(\mu-1)/\hat{c}_{\min}}$ is $O(\hat{c}_{\min} \cdot D^{2(\mu-1)/\hat{c}_{\min}} \cdot (D^{1/\hat{c}_{\min}} - 1))$-competitive.*

Proof. Let $\mathcal{O}$, $\mathcal{O}'$, and $\mathcal{B}$ be defined as in the proof of Theorem 1. Consider any request $r \in \mathcal{O}'$. Since r was not accepted by the weighted BGA, it must hold for the set $S(r)$ taken by OPT that (1) $S(r)$ was infeasible or (2) $f(S(r)) > W$ at the point where the BGA considered r. We capture these two cases via a potential function $\phi(S(r)) = \sum_{v \in S(r)} \sum_{i=0}^{\mu(v)-1} \phi(v, i)$ with

$$\begin{aligned}
\phi(v,i) &= \frac{\hat{c}_{\min}}{\hat{c}(v)} \sum_{j=-\mu(v)+i+1}^{\hat{c}(v)-c(v)-\mu(v)+i} D^{j/\hat{c}(v)} = \frac{\hat{c}_{\min}}{\hat{c}(v)} \cdot \frac{1}{D^{(\mu(v)-1)/\hat{c}(v)}} \sum_{j=0}^{\hat{c}(v)-c(v)+i-1} D^{j/\hat{c}(v)} \\
&\geq \frac{\hat{c}_{\min}}{\hat{c}(v)} \cdot \frac{D^{i/\hat{c}(v)}}{D^{(\mu(v)-1)/\hat{c}(v)}} \cdot \frac{D^{\hat{c}(v)-c(v)} - 1}{D^{1/\hat{c}(v)} - 1} \\
&\geq \frac{1}{D^{(\mu-1)/\hat{c}_{\min}}} \cdot \frac{1}{(D^{1/\hat{c}_{\min}} - 1)} \cdot D^{i/\hat{c}(v)} (f_v(\hat{c}(v) - c(v)) - 1) \\
&= \frac{1}{D^{(\mu-1)/\hat{c}_{\min}}} \cdot \frac{1}{(D^{1/\hat{c}_{\min}} - 1)} \cdot (f_v(\hat{c}(v) - c(v) + i) - D^{i/\hat{c}(v)})
\end{aligned}$$

In case 1, there must be a node $v \in S(r)$ with $c(v) < \mu(v)$ and therefore there is some i in $\sum_{i=0}^{\mu(v)-1} \phi(v,i)$ where $\hat{c}(v) - c(v) + i = \hat{c}(v)$. Thus,

$$\begin{aligned}\phi(S(r)) &\geq \sum_{i=0}^{\mu(v)-1} \phi(v,i) \geq \frac{1}{D^{(\mu-1)/\hat{c}_{\min}}} \cdot \frac{1}{(D^{1/\hat{c}_{\min}}-1)}(f_v(\hat{c}(v)) - D^{(\mu-1)/\hat{c}_{\min}}) \\ &\geq \frac{1}{D^{(\mu-1)/\hat{c}_{\min}}} \cdot \frac{D - D^{(\mu-1)/\hat{c}_{\min}}}{(D^{1/\hat{c}_{\min}}-1)}\end{aligned}$$

In case 2, we see that

$$\begin{aligned}\phi(S(r)) &= \sum_{v \in S(r)} \sum_{i=0}^{\mu(v)-1} \phi(v,i) \\ &\geq \frac{1}{D^{(\mu-1)/\hat{c}_{\min}}} \cdot \frac{1}{(D^{1/\hat{c}_{\min}}-1)} \sum_{v \in S(r)} \sum_{i=0}^{\mu(v)-1} (f_v(\hat{c}(v) - c(v) + i) - D^{i/\hat{c}_{\min}}) \\ &\geq \frac{1}{D^{(\mu-1)/\hat{c}_{\min}}} \cdot \frac{1}{(D^{1/\hat{c}_{\min}}-1)}(f(S(r)) - D^{1+(\mu-1)/\hat{c}_{\min}}) \\ &\geq \frac{1}{D^{(\mu-1)/\hat{c}_{\min}}} \cdot \frac{D - D^{(\mu-1)/\hat{c}_{\min}}}{(D^{1/\hat{c}_{\min}}-1)}\end{aligned}$$

Thus, in both cases,

$$\phi(S(r)) \geq \frac{1}{D^{\mu/\hat{c}_{\min}}} \cdot \frac{D - D^{(\mu-1)/\hat{c}_{\min}}}{(D^{1/\hat{c}_{\min}}-1)}$$

Note that if $c(v) = \hat{c}(v)$, i.e., v has not been used by any request or its capacity has been fully recharged in the meantime, $\phi(v,i) = 0$ for all i. Thus, whenever $\phi(v,i) > 0$, this is witnessed by requests whose BGA-sets include v. For the charging argument, for every slot occupied by a request $r' \in \mathcal{B}$ in v,

$$\hat{c}(v) \cdot \frac{\hat{c}_{\min}}{\hat{c}(v)} f_v(\hat{c}(v) - c_{r'}(v)) + \frac{\hat{c}_{\min}}{\hat{c}(v)} \Big(\sum_{i=0}^{\hat{c}(v)-c_{r'}(v)-1} f_v(i) \Big) + \frac{\hat{c}_{\min}}{\hat{c}(v)} \sum_{i=0}^{\mu-1} D^{-i/\hat{c}(v)}$$

has to be used, where the last term is due to the fact that $\phi(S(r))$ may also contain $D^{k/\hat{c}(v)}$'s with negative k's, which is covered by the request currently responsible for the lowest capacity slot in v. Note that

$$\frac{\hat{c}_{\min}}{\hat{c}(v)} \sum_{i=0}^{\mu-1} D^{-i/\hat{c}(v)} \leq \frac{\hat{c}_{\min}}{\hat{c}(v)(1 - 1/D^{1/\hat{c}(v)})} \leq \frac{\hat{c}_{\min} D^{1/\hat{c}(v)}}{\ln D} \leq \frac{\hat{c}_{\min} D^{1/\hat{c}_{\min}}}{\ln D} f(\hat{c}(v) - c_{r'}(v))$$

Continuing like in the proof of Theorem 1 will then result in the theorem. ◀

5 Conclusion

In this paper, we introduced the D-SPaRC problem and showed that simple greedy algorithms have a low competitive ratio for it and its extensions. We also extend the greedy approach to an online algorithm with predictions for the D-SPaRC problem with $\hat{c}_{\min} = \Omega(\log D)$ that is never worse than the greedy algorithm without predictions and can reach a constant competitive ratio in case of optimal predictions. The problem is further generalized to address both non-uniform request priorities and requests with non-uniform weights.

In the future, it will be interesting to investigate various extensions to our formulation. For example, strengthening the analysis to accommodate temporary modifications to the maximum capacity ($\hat{c}(v)$) of any involved resource v, potentially resulting from maintenance and future upgrades. It might also be worth exploring a modification where, after accepting one request, it can adaptively be updated — midway to use a different set of resources while still satisfying the D cap — to accommodate new incoming requests, thereby maximizing the total number of requests satisfied. In applications such as drone delivery, further improvements of the competitive ratio might be possible by allowing delayed decisions or load balancing via service drones.

References

1 Spyros Angelopoulos and Shahin Kamali. Rényi-ulam games and online computation with imperfect advice. In *MFCS*, volume 272 of *LIPIcs*, pages 13:1–13:15, 2023. doi:10.4230/LIPIcs.MFCS.2023.13.

2 Baruch Awerbuch, Yossi Azar, and Serge Plotkin. Throughput-competitive on-line routing. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 32–40. IEEE, 1993. doi:10.1109/SFCS.1993.366884.

3 Jakub Balabán, Matthias Gehnen, Henri Lotze, Finn Seesemann, and Moritz Stocker. Online knapsack problems with estimates. In *50th International Symposium on Mathematical Foundations of Computer Science, MFCS*, volume 345 of *LIPIcs*, pages 12:1–12:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPICS.MFCS.2025.12.

4 Patrick Bennett and Alan M. Frieze. On the chromatic number of random regular hypergraphs. *SIAM J. Discret. Math.*, 38(2):1369–1380, 2024. doi:10.1137/22M1544476.

5 Hans-Joachim Böckenhauer, Juraj Hromkovič, Dennis Komm, Peter Rossmanith, and Moritz Stocker. A survey of online knapsack problems. *Discrete Applied Mathematics*, 378:492–507, 2026. doi:10.1016/j.dam.2025.08.011.

6 Joan Boyar, Lene M. Favrholdt, Christian Kudahl, Kim S. Larsen, and Jesper W. Mikkelsen. Online algorithms with advice: A survey. *ACM Computing Surveys (CSUR)*, 50(2):1–34, 2017. doi:10.1145/2993749.2993766.

7 Joan Boyar, Lene M. Favrholdt, and Kim S. Larsen. Online unit profit knapsack with untrusted predictions. In *SWAT*, volume 227 of *LIPIcs*, pages 20:1–20:17. Schloss Dagstuhl - Leibniz-Zentrum f"ur Informatik, 2022. doi:10.4230/LIPIcs.SWAT.2022.20.

8 Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing. *Mathematics of Operations Research*, 34(2):270–286, 2009. doi:10.1287/moor.1080.0363.

9 Marek Cygan. Improved approximation for 3-dimensional matching via bounded pathwidth local search. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 509–518. IEEE Computer Society, 2013. doi:10.1109/FOCS.2013.61.

10 Mohammadreza Daneshvaramoli, Helia Karisani, Adam Lechowicz, Bo Sun, Cameron N. Musco, and Mohammad Hajiesmaili. Near-optimal consistency-robustness trade-offs for learning-augmented online knapsack problems. In *ICML*, volume 267 of *Proceedings of Machine Learning Research*, pages 12459–12489. PMLR, 13–19 Jul 2025. URL: https://proceedings.mlr.press/v267/daneshvaramoli25a.html.

11 George B. Dantzig and John H. Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959. doi:10.1287/mnsc.6.1.80.

12 Yuval Emek, Magnús M Halldórsson, Yishay Mansour, Boaz Patt-Shamir, Jaikumar Radhakrishnan, and Dror Rawitz. Online set packing and competitive scheduling of multi-part tasks. In *Proceedings of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 440–449, 2010. doi:10.1145/1835698.1835800.

13 Yuval Emek, Magnús M Halldórsson, Yishay Mansour, Boaz Patt-Shamir, Jaikumar Radhakrishnan, and Dror Rawitz. Online set packing. *SIAM Journal on Computing*, 41(4):728–746, 2012. doi:10.1137/110820774.

14 Paul Erdős and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In *Infinite and Finite Sets (Colloq. Math. Soc. J. Bolyai, Vol. 11)*, pages 609–627. North-Holland, 1975.

15 Martin Fürer and Huiwen Yu. Approximating the k -set packing problem by local improvements. In *Combinatorial Optimization - Third International Symposium, ISCO*, volume 8596 of *Lecture Notes in Computer Science*, pages 408–420. Springer, 2014. doi:10.1007/978-3-319-09174-7_35.

16 Matthias Gehnen, Henri Lotze, and Peter Rossmanith. Online simple knapsack with bounded predictions. In *STACS*, volume 289 of *LIPIcs*, pages 37:1–37:20. Schloss Dagstuhl - Leibniz-Zentrum f"ur Informatik, 2024. doi:10.4230/LIPIcs.STACS.2024.37.

17 Mohammad H. Hajiesmaili. Trade-off analysis in learning-augmented algorithms with societal design criteria. *SIGMETRICS Perform. Evaluation Rev.*, 51(2):53–58, 2023. doi:10.1145/3626570.3626590.

18 Magnús M. Halldórsson, Jan Kratochvíl, and Jan Arne Telle. Independent sets with domination constraints. *Discret. Appl. Math.*, 99(1-3):39–54, 2000. doi:10.1016/S0166-218X(99)00124-9.

19 Elad Hazan, Shmuel Safra, and Oded Schwartz. On the complexity of approximating k-set packing. *Comput. Complex.*, 15(1):20–39, 2006. doi:10.1007/S00037-006-0205-6.

20 Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. Online knapsack with frequency predictions. In *NeurIPS*, pages 2733–2743, 2021. URL: https://dl.acm.org/doi/10.5555/3540261.3540470.

21 Suho Kang, Ziyang Liu, and Rajan Udwani. A black-box approach for exogenous replenishment in online resource allocation. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 326–340. Springer, 2025. doi:10.1007/978-3-031-93112-3_24.

22 Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103, 1972. doi:10.1007/978-1-4684-2001-2_9.

23 Petr Kolman and Christian Scheideler. Improved bounds for the unsplittable flow problem. *Journal of Algorithms*, 61(1):20–44, 2006. doi:10.1016/j.jalgor.2004.07.006.

24 Dennis Komm. *An introduction to online computation*. Springer, 2016. URL: https://link.springer.com/book/10.1007/978-3-319-42749-2.

25 Gilbert Laporte. Fifty years of vehicle routing. *Transportation science*, 43(4):408–416, 2009. doi:10.1287/trsc.1090.0301.

26 Adam Lechowicz, Rik Sengupta, Bo Sun, Shahin Kamali, and Mohammad Hajiesmaili. Time fairness in online knapsack problems. In *ICLR*, 2024. URL: https://openreview.net/forum?id=9kG7TwgLYu.

27 Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *Journal of the ACM (JACM)*, 68(4):1–25, 2021. doi:10.1145/3447579.

28 Giusy Macrina, Luigi Di Puglia Pugliese, Francesca Guerriero, and Gilbert Laporte. Drone-aided routing: A literature review. *Transportation Research Part C: Emerging Technologies*, 120:102762, Nov 2020. doi:10.1016/j.trc.2020.102762.

29 Alberto Marchetti-Spaccamela and Carlo Vercellis. Stochastic on-line knapsack problems. *Mathematical Programming*, 68:73–104, 1995. doi:10.1007/BF01585758.

30 Thorben Tröbst and Rajan Udwani. Almost tight bounds for online hypergraph matching. *Operations Research Letters*, 55:107143, 2024. doi:10.1016/j.orl.2024.107143.

31 Ali Zeynali, Bo Sun, Mohammad Hassan Hajiesmaili, and Adam Wierman. Data-driven competitive algorithms for online knapsack and set cover. In *AAAI*, pages 10833–10841. AAAI Press, 2021. doi:10.1609/AAAI.V35I12.17294.

32 Yunhong Zhou, Deeparnab Chakrabarty, and Rajan M. Lukose. Budget constrained bidding in keyword auctions and online knapsack problems. In Christos H. Papadimitriou and Shuzhong Zhang, editors, *Internet and Network Economics, 4th International Workshop, WINE 2008, Shanghai, China, December 17-20, 2008. Proceedings*, Lecture Notes in Computer Science, pages 566–576. Springer, 2008. doi:10.1007/978-3-540-92185-1_63.