Optimal Competitive Ratio for Optimization Problems with Congestion Effects

Miriam Fischer
Imperial College, London, UK

Dario Paccagnan
Imperial College, London, UK

Cosimo Vinci
University of Salento, Lecce, Italy

Abstract

In this work we study online optimization problems with congestion effects. These are problems where tasks arrive online and a decision maker is required to allocate them on the fly to available resources in order to minimize the cost suffered, which grows with the amount of resources used. This class of problems corresponds to the online counterpart of well-known studied problems, including optimization problems with diseconomies of scale [33], minimum cost in congestion games [25], and load balancing problems [4].

Within this setting, our work settles the problem of designing online algorithms with *optimal competitive ratio*, i.e., algorithms whose incurred cost is as close as possible to that of an oracle with complete knowledge of the future instance ahead of time. We provide three contributions underpinning this result. First, we show that no online algorithm can achieve a competitive ratio below a given factor depending solely on the resource costs. Second, we show that, when guided by carefully modified cost functions, the greedy algorithm achieves a competitive ratio matching this lower bound and thus is optimal. Finally, we show how to compute such modified cost functions in polynomial time.

2012 ACM Subject Classification Theory of computation → Online algorithms; Mathematics of computing → Mathematical optimization; Theory of computation → Algorithmic game theory

Keywords and phrases Online Algorithms, Competitive Ratio, Algorithmic Game Theory, Greedy Algorithms, Congestion Games

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2025.9

Category APPROX

Funding This work is partially supported by: a Google DeepMind Scholarship awarded to Miriam Fischer; the PNRR MIUR project FAIR – Future AI Research (PE00000013), Spoke 9 – Green-aware AI; MUR – PNRR IF Agro@intesa; the Project SERICS (PE00000014) under the NRRP MUR program funded by the EU – NGEU; GNCS-INdAM.

Acknowledgements We thank the reviewers for valuable comments and suggestions.

1 Introduction

In this work, we study optimization problems with *congestion* effects. These are cost minimization problems where the goal is to allocate resources to tasks, and the cost suffered by the system grows with the amount of tasks assigned to each resource. This fundamental class of problems is ubiquitous, including celebrated classes of problems such as load balancing [4, 43, 32, 16], social cost minimization in congestion games [40, 35, 25], online virtual circuit routing [3, 31], as well as energy-efficient algorithm design [1, 2], for which we now have a comprehensive picture of the achievable performances and fundamental algorithmic barriers.

However, while much of the existing literature focuses on offline settings, real-world decisions often need to be taken sequentially, with no access to future information, and irrevocably. For these reasons, we are interested in an *online* setting, where tasks arrive sequentially and resources need to be allocated on the fly. In this context, one measures the quality of a given algorithm over a class of inputs through the notion of *competitive ratio* [42, 27]. The latter represents the worst-case ratio between the cost achieved by the given algorithm and the optimal cost of an offline algorithm with complete knowledge of the input ahead of time. The goal is then to design an algorithm that minimize such measure of inefficiency.

In this work we settle this problem and develop algorithms that are optimal in the sense that they achieve minimal competitive ratio. We do so by developing a greedy algorithm that is guided in its search by an auxiliary cost function that, crucially, does not align with the cost one wishes to minimize in the first place.

We note that, prior to our work, results on the design of online algorithms for specialized subclasses of problems were given in [4, 16, 15] for the widely-studied class of load balancing problems. Within this setting, these works analyze the performance of standard greedy algorithms whose decisions are guided by the original cost function, rather than by general auxiliary functions, as in our more general algorithmic framework. However, their approach has been shown to be optimal only for specific settings, such as those involving polynomial resource cost functions and weighted tasks. The design of optimal online algorithms for other settings, such as those involving unweighted tasks and/or potentially more general cost functions than polynomials, remains an open problem - see the first paragraph of Section 1.1 for further details. In this respect, our work resolves a number of open questions, e.g., by providing the optimal competitive ratio, and improves upon previous design, e.g., by developing a general algorithmic framework applicable to a variety of settings with strong guarantees of both optimality and computational efficiency. At the same time, we highlight that the results we provide apply *well-beyond* the class of load balancing problems and extend to general resource cost functions. We are not aware of optimal algorithms at this level of generality. As for offline counterparts of the problem we consider, our work connects with the contributions of [33] and [25], which provide tight polynomial time algorithms for optimization problems with diseconomies of scale, and for minimizing social cost in congestion games. Our work settles a similar question to those of the above-cited papers but, crucially, in the *online* setting.

Finally, we highlight that our results can also be interpreted under a game-theoretic lens, wherein greedy agents appear online, and one is tasked with introducing *interventions*, in the form of incentives or taxes, to steer their behavior and reduce the resulting cost. In this context, our work provides a recipe to compute *online optimal interventions*, i.e., interventions that minimize the inefficiency of the resulting allocation. Indeed, the auxiliary cost functions we design for the greedy algorithm can be leveraged to define incentives or taxes to influence the agents' behavior. This is discussed in detail in Section 6.

Our contribution and techniques

Our work settles the problem of designing optimal algorithms for online optimization problems with congestion effects. Three technical contributions support this claim.

- First, we provide a lower bound on the competitive ratio of *any* online algorithm. This lower bound provides an insurmountable barrier on the achievable performance and, interestingly, depends only on the class of resource costs used, e.g., polynomials. We obtain this result for any class of non-decreasing and non-negative cost functions.

- Second, we show that, for strictly convex cost functions, the greedy algorithm (guided by carefully modified cost functions) achieves a competitive ratio matching this lower bound and thus is optimal.
- Third, we give a constructive approach to compute the modified cost functions to be used by the greedy algorithm, through poly-time solvable linear programs. In particular, we give an explicit construction for the case of polynomial resource costs.

Before discussing our techniques, we highlight that, while optimal, the algorithm we propose is easily and efficiently implementable. In fact, the modified cost functions employed by the greedy algorithm do not need to be computed online but can instead be evaluated offline. This aspect is important as it ensures that the online component of the algorithm incurs small computational cost, consisting in the use of a simple greedy algorithm with respect to the previously computed modified costs.

Techniques. Our work utilizes a mix of combinatorial and duality-based ideas. Regarding the lower bound, perhaps interesting is the fact that the instance we construct features resources with *identical* cost functions, thus implying that the fundamental barrier we obtain on the achievable competitive ratio is very strong, as it applies already in simple settings such as that of online load balancing problems – see first paragraph of Section 1.1 for further details. The core idea in deriving our lower bound is to make the constructed instance algorithm-dependent in a way so that tasks arrive in groups. Within each group, any possible allocation of tasks to resources is constructed to incur the same cost, so that no online algorithm can distinguish between the quality of one allocation from another.

Designing an optimal algorithm with matching upper bound requires, instead, a different line of work. We follow here a three-pronged approach. First, we leverage linear programming to tightly evaluate the competitive ratio of the greedy algorithm with respect to a-priori given costs, though in a setting where such costs need not coincide with the objective we wish to minimize. Second we utilize linear programming duality to design optimal costs, i.e., costs that, once employed by greedy, minimize the competitive ratio. Third, we show that the optimal value of such linear program matches the lower bound previously found. Finally, as the resulting linear programs contain infinitely many constraints, we show how only finitely many of them can be considered to provide an explicit construction of the modified costs.

1.1 Related work

Our work relates to a number of research streams, which we review in the following paragraphs.

Online load balancing. In online load balancing problems, tasks arrive sequentially over time, and the decision maker needs to schedule on the fly each individual task to a *single* machine from a given set of machines, so as to minimize the long-term cost incurred (typically measured by the p-norm of the machine loads or, in a closely related formulation, by polynomial resource cost functions). This is a well-studied area, starting with the works of [4, 5, 6] and various follow-up contributions, e.g., [10, 15, 16, 19, 36, 43]. In the weighted version of load balancing, the increase in cost that a task causes to a resource depends on a task-specific weight. For this setting, a specialized greedy algorithm has been shown to guarantee the best competitive ratio when the resource cost functions are restricted to polynomials [15, 36]. However, such an optimality guarantee does not apply to more general cost functions or to the unweighted version of the problem (where all task weights are unitary, and thus potentially allow a lower competitive ratio), which, as will be clear from

Section 2, belongs to the class of online optimization problems studied in this manuscript. In such unweighted settings, the works of [10, 16, 19, 22, 43] provide upper and lower bounds for *specialized* classes of online algorithms. Contrary to that, our work settles the general question of designing optimal online algorithms and deriving fundamental barriers, in the form of tight lower bounds, to which *any* online algorithm is subject and which depend only on the considered class of resource cost functions.

Perhaps interestingly, the instance we construct to lower bound the competitive ratio for the broader class of optimization problems with congestion effects, is also an instance of online load balancing problems, thus giving a tight answer for this smaller, but important, class too.

Online congestion games. The setup considered in online congestion games is similar to that of online load balancing problems, with the crucial difference that individual tasks (referred to as agents in that literature) may require more then one resource for their completion. This class of problems is subsumed, too, by the broader class of online optimization problems we study in this work, motivating the ensuing comparison. Most of the literature in online congestion games has focused on providing performance certificates, in the form of competitive ratios, for fixed classes of online algorithms, notably greedy algorithms [4, 8, 9, 10, 15, 16, 19, 22, 29, 43, 44]. Specifically, for linear latency functions (see Section 4 for its definition), [19] and [9] provide a tight competitive ratio of $\frac{(\phi+1)^2}{\phi} \approx 4.236$, where $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio. For quadratic and cubic latency functions, [8] provides upper bounds of approximately 37.589 and 527.323, respectively. A general lower bound of $(d+1)^{d+1}$ is given by [22] for polynomial latency functions of degree d, while upper bounds for the same setting are shown in [29]. The above-mentioned works analyze the performance of greedy algorithms utilizing the original cost function. Instead, in this work we provide tight lower bounds that hold for *any* online algorithm, and further show that one can achieve such tight lower bound by judiciously modifying the cost function used by the greedy algorithm. To the best of our knowledge, lower bounds that apply to *any* online algorithm were previously only known for linear latency functions [26]. Our work not only improves upon the lower bound in [26, Proposition 4.4] by a factor of 2, but - more importantly - provides lower bounds for arbitrary classes of non-negative and non-decreasing latency functions. Finally, we highlight that, while [11] seemingly improves on the performance achieved by the above-cited works, their algorithm requires a priori knowledge of an optimal allocation and of the full instance. As such, their results are not applicable in an online setting.

Offline optimization problems with congestion effects. Offline versions of the problem we consider have been well studied, both under the umbrella of *optimization problems (or generalized network design problems) with diseconomies of scale* [33, 21] and under that of *minimum social cost in congestion games* [25]. While there is significant work in both these domains including [35, 2], notable are the works of [33, 21, 25], which provide polynomial time algorithms with optimal approximation and matching lower bounds, and apply these results to a variety of relevant cost functions, including polynomial ones. Our work settles a similar question to that of the above-cited papers, but crucially, in the *online* setting.

Online optimization. Finally, we would like to highlight that our work focuses on developing online algorithms in the spirit of competitive analysis [13, 14, 23], i.e., in a setting where one compares the performance of the algorithm at hand with that of an offline optimal algorithm with complete knowledge of the future instance. While connected, this approach differs significantly from other types of analysis performed in online settings, e.g., the extensive line of work on no-regret algorithms [41, 34].

Organization of the manuscript

The remainder of the manuscript is organized as follows. In Section 2, we present the model. In Section 3, we provide a lower bound on the competitive ratio of any online algorithm. In Section 4, we introduce a greedy algorithm that employs modified cost functions, and show that this algorithm achieves best-possible competitive ratio among all online algorithms. In Section 5, we address the efficient computation of such modified cost functions. Finally, in Section 6, we outline connections with Game Theory and, specifically, how our results can be applied to the design of optimal interventions in congestion games. All omitted results and proofs are deferred to the appendix.

2 Problem formulation

We consider an online version of a classical cost minimization problem that involves resource selection and congestion, referred to as *online optimization with congestion effects*. With regards to notation, $\mathbb{N}$ denotes the set of natural numbers, $\mathbb{N}_0$ denotes the set of natural numbers including zero, and given an integer $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \dots, n\}$ and $[n]_0$ the set $\{0, 1, \dots, n\}$. In a specific instance of this setting, we are given a fixed set of resources $\mathcal{R}$, while individual tasks arrive at discrete time steps $t \in [T]$, each with a feasible set $\mathcal{A}_t \subseteq 2^{\mathcal{R}} \setminus \{\emptyset\}$ containing collections of resources needed to complete such task. The length of the time horizon $T \in \mathbb{N}$ is unknown in advance and, for simplicity, we identify each task with the time step t at which it arrives. At each time step t, the decision maker needs to irrevocably assign the corresponding task to a non-empty collection of resources from the set $\mathcal{A}_t$, which we denote with $a_t \in \mathcal{A}_t$. We refer to the collection of resources a_t to which task t is assigned as the *action* for task t, and $\mathcal{A}_t$ denotes the set of admissible actions for task t. An *allocation* $a = (a_1, \dots, a_T)$ specifies the resources allocated to each task throughout the whole horizon. Each resource $r \in \mathcal{R}$ is associated with a non-negative and non-decreasing *cost function* $c_r(x_r) : \mathbb{N}_0 \to \mathbb{R}_{\geq 0}$, with $c_r(0) = 0$ and $c_r(x) > 0$ for $x > 0$[1], that relates the number of tasks assigned to each resource r (in a given allocation) to the cost incurred by r due to the assignment. An example of a well-studied cost functions is that of *polynomials* with non-negative coefficients and maximum degree d. The *total cost* incurred at the end of the time horizon and associated to the allocation $a = (a_1, \dots, a_T)$ is then obtained by summing the costs experienced over all resources, i.e.,

$$C(a) = \sum_{r \in \mathcal{R}} c_r(|a|_r)$$

where $|a|_r \coloneqq |\{t \in [T] : r \in a_t\}|$ denotes the *congestion* of resource r in a, that is, the number of tasks assigned to resource r, under allocation a.

In this work, we are interested in developing online algorithms that minimize the total cost. Specifically, an *online algorithm* Alg acting on instance $I = (\mathcal{R}, \{c_r\}_{r \in \mathcal{R}}, \{\mathcal{A}_t\}_{t=1}^T)$ is a map that irrevocably and deterministically[2] associates to each time/task t a feasible action $a_t \in \mathcal{A}_t$ solely using information available up to that time, i.e., solely using knowledge of $\mathcal{R}$, $\{c_r\}_{r \in \mathcal{R}}$ and of the previous decisions $\{a_1, \dots, a_{t-1}\}$. We denote the allocation $(a_1, \dots, a_T)$ generated by algorithm Alg as $\textsc{Alg}(I)$. Finally, we evaluate the quality of an online algorithm by

[1] For our purposes, the assumption that $c_r(x) > 0$ for $x > 0$ is without loss of generality. Indeed, it can be shown that if this condition does not hold, the competitiveness of online solutions become arbitrarily bad (see Remark 15 in the appendix for further details).

[2] We focus in this work on deterministic algorithms.

considering its *competitive ratio*, that is, the worst-case ratio between the cost achieved by this algorithm and the optimal cost that could have been achieved by an algorithm that minimizes the cost $C(a)$ with complete knowledge of the instance ahead of time. Formally, given a class of input instances $\mathcal{I}$, the *competitive ratio* of algorithm ALG is defined as

$$\sup_{I \in \mathcal{I}} \frac{C(\textsc{Alg}(I))}{C(\textsc{Opt}(I))},$$

where $\textsc{Opt}(I)$ denotes an optimal offline solution, i.e., $\textsc{Opt}(I) \in \arg\min_{a \in \mathcal{A}_1 \times \cdots \times \mathcal{A}_T} C(a)$.

In this work we aim at designing algorithms that minimize such competitive ratio, i.e., whose corresponding cost is as close as possible to that of an offline optimal solution. In our analysis, we will often consider classes of instances that are generated using resource costs taken from a given set of resource cost functions $\mathcal{C}$, e.g., polynomials. These are instances where each resource cost c_r belongs to the given set $\mathcal{C}$. From now on, we implicitly assume that all sets $\mathcal{C}$ of cost functions under consideration, by default, satisfy the properties stated at the beginning of the section (e.g., non-negativity and non-decreasing monotonicity); additional assumptions (e.g., strict convexity) will be specified explicitly. Interesting sub-classes of the model are given by *load balancing instances*, where each feasible action in $\mathcal{A}_t$ contains a *single* resource, and *instances with identical resources*, where all resources have the same cost function, i.e., $c_r = c_{r'}$ for all $r, r' \in \mathcal{R}$.

3 Lower Bound on Competitive Ratio

In this section, we provide a lower bound on the competitive ratio of any online algorithm, parametrized by the class of resource costs $\mathcal{C}$ utilized in constructing the instances. Towards this goal, given a cost function $c \in \mathcal{C}$, it is instrumental to introduce

$$\mathsf{CR}(c) := \sup_{y \in \mathbb{N}_0} \sup_{(x_0,\ldots,x_y) \in \mathbb{N}^{y+1}} \frac{\sum_{i=0}^{y} c(i) \cdot \frac{1}{x_i} \prod_{h=1}^{i} \frac{x_h}{x_h+1} + c(y+1) \prod_{h=1}^{y} \frac{x_h}{x_h+1}}{\sum_{i=0}^{y} c(x_i) \cdot \frac{1}{x_i} \prod_{h=1}^{i} \frac{x_h}{x_h+1}}. \tag{1}$$

As will become apparent later, $\mathsf{CR}(c)$ arises from the lower bound instance used in the proof of Theorem 1. We observe that $\mathsf{CR}(c) \geq 1$, since setting $y = 0$ and $(x_0) = (1)$ in the right-hand side of (1) yields 1, regardless of the choice of the cost function c. The following theorem constitutes our first main result. It shows that no online algorithm can achieve a competitive ratio below $\sup_{c \in \mathcal{C}} \mathsf{CR}(c)$.

▶ **Theorem 1.** *For an arbitrary class $\mathcal{C}$ of cost functions, no online algorithm can achieve a competitive ratio over instances generated by $\mathcal{C}$ below $\sup_{c \in \mathcal{C}} \mathsf{CR}(c)$. This result also holds when restricting to load balancing instances with identical resources.*

Proof. We will show that, for any fixed cost function $c \in \mathcal{C}$ and online algorithm ALG, there exists a load balancing instance I where all cost functions are identically equal to c and such that $\frac{C(\textsc{Alg}(I))}{C(\textsc{Opt}(I))} \geq \mathsf{CR}(c)$. By taking the worst case over $c \in \mathcal{C}$, it follows that $\sup_{c \in \mathcal{C}} \mathsf{CR}(c)$ provides a lower bound on the competitive ratio of any online algorithm over load balancing instances with identical resources and cost functions drawn from $\mathcal{C}$. One then concludes by observing that such a lower bound applies directly to optimization problems with congestion effects, as these form a superset of the set of load balancing instances considered.

Fix a cost function $c \in \mathcal{C}$, and an online algorithm ALG. To prove the claim, we design an instance constructed adversarially with respect to any choice made by ALG at each task's arrival, aiming to obtain an allocation with the highest possible cost. Specifically, for a given $y \in \mathbb{N}_0$ and a sequence $(x_h)_{h \in [y]_0}$ of $y+1$ positive integers, let $I = I(y, (x_h)_{h \in [y]_0}, c, \textsc{Alg}) = (\mathcal{R}, \{c_r\}_{r \in \mathcal{R}}, \{\mathcal{A}_t\}_{t=1}^{T})$ be a load balancing instance defined as follows:

- *Resources and Cost Functions:* Let $p := \prod_{h=0}^{y} x_h$. $\mathcal{R}$ is a set of $p \prod_{h=0}^{y} \frac{x_h+1}{x_h} = \prod_{h=0}^{y}(x_h+1)$ distinct resources. The cost function c_r of each resource $r \in \mathcal{R}$ is identically equal to c.
- *Time Horizon:* Given $i \in [y+1]$, let $q_i := p \prod_{h=i}^{y} \frac{x_h+1}{x_h}$; we observe that, by the choice of p, $|\mathcal{R}|$ and each q_i are integers. We set $T := \sum_{i=1}^{y+1} q_i$. We partition the set of time-steps/tasks $[T]$ into $y+1$ sets $Q_1, \dots, Q_y, Q_{y+1}$ of consecutive time steps, where each Q_i is defined as $\{1 + \sum_{j=1}^{i-1} q_j, \dots, \sum_{j=1}^{i} q_j\}$ (i.e., Q_i is made of q_i consecutive time steps).
- *Tasks and Actions:* According to the definitions of $Q_1, \dots, Q_{y+1}$, the algorithm ALG first processes all the tasks in Q_1, then all the tasks in Q_2, and so forth. As the instance is of load balancing type, we can characterize the feasible set of actions $\mathcal{A}_t$ for each task t as $\mathcal{A}_t = \{\{r\} \mid r \in \mathcal{R}_t\}$, for a given set $\mathcal{R}_t \subseteq \mathcal{R}$ of singletons to which each task t can be assigned, constructed as follows. Given $i \in [y+1]$ and $t \in Q_i$, $\mathcal{R}_t$ is adversarially and iteratively defined, based on the decisions that the algorithm would take up to and including time $t-1$. Specifically, for $t \in Q_i$, assuming that the algorithm has allocated the first $t-1$ tasks, $\mathcal{R}_t$ consists of all resources which have a congestion of $i-1$ at time t (that is, just before processing task t). This design procedure is feasible, as the considered online algorithm is deterministic, and its decision at time t is uniquely determined by the previous observations and decisions.

The following lemma exhibits some fundamental structural properties of the set of resources $\mathcal{R}_t$ to which each task t can potentially be assigned; notably, it shows that each $\mathcal{R}_t$ is non-empty and well-defined.

▶ **Lemma 2.** *There exists a sequence of $y+2$ non-empty resource subsets $\mathcal{S}_0, \mathcal{S}_1, \dots, \mathcal{S}_{y+1} \subseteq \mathcal{R}$, with $\mathcal{R} = \mathcal{S}_0 \supset \mathcal{S}_1 \supset \dots \supset \mathcal{S}_{y+1}$ and $|\mathcal{S}_i| = |Q_i| = q_i$ for any $i \in [y+1]$, such that ALG assigns distinct tasks $t \in Q_i$ to distinct resources $r \in \mathcal{S}_i$ in a one-to-one correspondence. Furthermore, for any $i \in [y+1]$ and task $t \in Q_i$, $\mathcal{R}_t$ consists of all the resources in $\mathcal{S}_{i-1}$ to which no task in Q_i has been previously assigned by ALG.*

The proof of the above lemma directly follows from the iterative and adversarial definition of each $\mathcal{R}_t$; in particular, it holds since, for any $i \in [y+1]$ and time step $t \in Q_i$, there always exists at least a resource in $\mathcal{S}_{i-1}$ having congestion $i-1$ at time t, to which task t can be assigned. Figure 1 represents an example of input instance I, along with the application of Lemma 2.

By exploiting Lemma 2 and the related notation, we can also efficiently characterize the total cost of the allocation returned by algorithm ALG, along with the resulting competitive ratio. Let $a^{\text{ALG}} := \text{ALG}(I)$ denote the allocation returned by ALG. We observe that, for any $i \in [y+1]_0$, the congestion of each resource in $\mathcal{S}_i \setminus \mathcal{S}_{i+1}$ is equal to i, where we set $\mathcal{S}_{y+2} := \emptyset$. Indeed, by Lemma 2, each resource in $\mathcal{S}_1 \subset \mathcal{S}_0 = \mathcal{R}$ first receives a task, followed by each resource in $\mathcal{S}_2 \subset \mathcal{S}_1$, and so on. Consequently, since $\mathcal{R} = \mathcal{S}_0 \supset \mathcal{S}_1 \supset \dots \supset \mathcal{S}_{y+1} \supset \mathcal{S}_{y+2} = \emptyset$, by the end of the algorithm, every resource in $\mathcal{S}_i \setminus \mathcal{S}_{i+1}$ is used by exactly i tasks. Then, the total cost achieved by a^{ALG} is

$$C(a^{\text{ALG}}) = \sum_{i=0}^{y+1} c(i) \cdot |\mathcal{S}_i \setminus \mathcal{S}_{i+1}| = \sum_{i=1}^{y} c(i) \cdot \frac{p}{x_i} \prod_{h=i+1}^{y} \frac{x_h+1}{x_h} + p \cdot c(y+1),$$

where the last equality holds by $|\mathcal{S}_i \setminus \mathcal{S}_{i+1}| = |Q_i| - |Q_{i+1}| = q_i - q_{i+1} = p \prod_{h=i}^{y} \frac{x_h+1}{x_h} - p \prod_{h=i+1}^{y} \frac{x_h+1}{x_h} = \frac{p}{x_i} \prod_{h=i+1}^{y} \frac{x_h+1}{x_h}$ for $i \in [y]_0$, and $|\mathcal{S}_{y+1} \setminus \mathcal{S}_{y+2}| = |\mathcal{S}_{y+1}| = p$.

Now, let $\tilde{a}$ denote the allocation that, for any $i \in [y+1]$, uniformly distributes all q_i tasks of Q_i over all resources in $\mathcal{S}_{i-1} \setminus \mathcal{S}_i$; $\tilde{a}$ will be used as benchmark to measure the performance of a^{ALG}. As a consequence of Lemma 2, $\mathcal{S}_{i-1} \setminus \mathcal{S}_i \subseteq \mathcal{R}_t$ is valid for any $i \in [y+1]$ and task

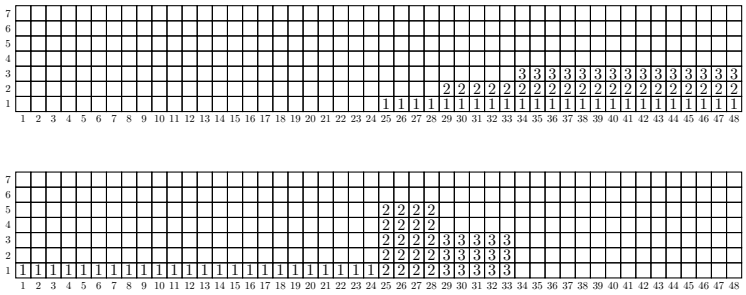

Figure 1 An example of an input instance I with $y = 2$ and $(x_0, x_1, x_2) = (1, 5, 3)$ derived from a given online algorithm ALG, along with the allocation a^{ALG} returned by the algorithm (upper figure) and the benchmark allocation $\tilde{a}$ (lower figure). In both the upper and lower figures, the indexes along the x-axis represent the $(x_0 + 1) \cdot (x_1 + 1) \cdot (x_2 + 1) = 48$ resources in $\mathcal{R}$. Each square labeled with $i \in [y + 1] = \{1, 2, 3\}$ in the upper (resp. lower) figure represents a distinct task of Q_i, and its position on the x-axis indicates the resource to which it is assigned in a^{ALG} (resp. $\tilde{a}$). The set $\mathcal{R}_t$ of feasible resources for each task $t \in [T] \in [59]$ is defined iteratively and adversarially as follows: the first task of Q_1 can potentially be assigned to each resource of $\mathcal{S}_0 = \mathcal{R}$, and all subsequent tasks of Q_1 can potentially be assigned to each resource of $\mathcal{S}_0$ to which no task in Q_1 has been previously assigned by ALG; then, the first task of Q_2 can potentially be assigned to each resource of $\mathcal{S}_1$ (resources 25 to 48), and all subsequent tasks of Q_2 can potentially be assigned to each resource of $\mathcal{S}_1$ to which no task in Q_2 has been previously assigned by ALG; finally, the above reasoning extends to Q_3.
The upper figure represents the allocation a^{ALG} returned by ALG. ALG assigns all tasks in Q_1, Q_2, Q_3 to the resources in $\mathcal{S}_1$ (resources 25 to 48), $\mathcal{S}_2$ (resources 29 to 48), $\mathcal{S}_3$ (resources 34 to 48), respectively, in a one-to-one-correspondence. At the end of the execution of ALG, the congestion of each resource in the obtained allocation a^{ALG} is given by the number of labeled squares (i.e., tasks) in the column corresponding to that resource. Specifically, each resource in $\mathcal{S}_0 \setminus \mathcal{S}_1$ (resources 1 to 24), $\mathcal{S}_1 \setminus \mathcal{S}_2$ (resources 25 to 28), $\mathcal{S}_2 \setminus \mathcal{S}_3$ (resources 29 to 33) and $\mathcal{S}_3$ (resources 34 to 48) has congestion 0,1,2 and 3 (under a^{ALG}), respectively.
The lower figure represents the benchmark allocation $\tilde{a}$. Specifically, x_i tasks of Q_i are assigned to each resource in $\mathcal{S}_{i-1} \setminus \mathcal{S}_i$. The congestion of each resource in allocation $\tilde{a}$ is again represented by the number of labeled squares in the column corresponding to that resource. Specifically, each resource in $\mathcal{S}_0 \setminus \mathcal{S}_1$, $\mathcal{S}_1 \setminus \mathcal{S}_2$, $\mathcal{S}_2 \setminus \mathcal{S}_3$ and $\mathcal{S}_3$ has congestion 1,5,3 and 0 (under $\tilde{a}$), respectively.

$t \in Q_i$ (i.e., each task in Q_i can potentially be assigned to each resource in $\mathcal{S}_{i-1} \setminus \mathcal{S}_i$), and then $\tilde{a}$ effectively represents a feasible allocation. Figure 1 represents the allocations a^{ALG} and $\tilde{a}$ in an example input instance I. Then, the congestion of each resource in $\mathcal{S}_{i-1} \setminus \mathcal{S}_i$ under allocation $\tilde{a}$ is $\frac{q_i}{|\mathcal{S}_{i-1} \setminus \mathcal{S}_i|} = \frac{p \prod_{h=i}^{y} (x_h+1)/x_h}{(p/x_{i-1}) \prod_{h=i}^{y} (x_h+1)/x_h} = x_{i-1}$. Thus, the total cost of allocation $\tilde{a}$ is

$$C(\tilde{a}) = \sum_{i=1}^{y+1} c(x_{i-1}) \cdot |\mathcal{S}_{i-1} \setminus \mathcal{S}_i| = \sum_{i=0}^{y} c(x_i) \cdot |\mathcal{S}_i \setminus \mathcal{S}_{i+1}| = \sum_{i=0}^{y} c(x_i) \cdot \frac{p}{x_i} \prod_{h=i+1}^{y} \frac{x_h + 1}{x_h},$$

where, in the second equality, we have shifted the index i by one. By exploiting the values of $C(a^{\text{ALG}})$ and $C(\tilde{a})$ determined above, we conclude that the competitive ratio of algorithm ALG, when applied to instance I, is

$$\begin{aligned} \frac{C(\text{ALG}(I))}{C(\text{OPT}(I))} \geq \frac{C(a^{\text{ALG}})}{C(\tilde{a})} &= \frac{\sum_{i=1}^{y} c(i) \frac{p}{x_i} \prod_{h=i+1}^{y} \frac{x_h+1}{x_h} + p \cdot c(y+1)}{\sum_{i=0}^{y} c(x_i) \frac{p}{x_i} \prod_{h=i+1}^{y} \frac{x_h+1}{x_h}} \\ &= \frac{\sum_{i=1}^{y} c(i) \frac{1}{x_i} \prod_{h=1}^{i} \frac{x_h}{x_h+1} + \prod_{h=1}^{y} \frac{x_h}{x_h+1} c(y+1)}{\sum_{i=0}^{y} c(x_i) \frac{1}{x_i} \prod_{h=1}^{i} \frac{x_h}{x_h+1}}, \end{aligned} \tag{2}$$

where the last equality in (2) is obtained multiplying by $\frac{1}{p}\prod_{h=1}^{y}\frac{x_h}{x_h+1}$ both numerator and denominator of the fraction in the previous line. Taking the worst case over $y \in \mathbb{N}_0$ and $(x_h)_{h\in[y]_0} \in \mathbb{N}^{y+1}$, one immediately obtains the expression of $\mathsf{CR}(c)$, and thus concludes. ◀

Theorem 1 can be applied to quantify lower bounds for specific classes of resource costs, such as polynomials.

▶ **Corollary 3.** *No online algorithm can achieve a competitive ratio lower than* $(d+1)^{d+1}$*, over instances generated by polynomial resource costs of maximum degree* $d+1$*, with* $d \geq 0$*. This result also holds when restricting to load balancing instances with identical monomial cost functions* $c(x) = x^{d+1}$*.*

The proof of the above corollary is deferred to Section A.

4 Optimal Competitive Ratio via Modified Greedy Algorithms

In Section 3 we show that the efficiency of online algorithms cannot exceed a certain barrier, established by the lower bound of $\sup_{c\in\mathcal{C}}\mathsf{CR}(c)$ on their competitive ratio, when cost functions are drawn from $\mathcal{C}$. Thus, it is interesting to ask a complementary question: does there exist an online algorithm whose competitive ratio matches this bound? Despite the fact that the class of instances used to obtain the lower bound $\sup_{c\in\mathcal{C}}\mathsf{CR}(c)$ is rather complex, in this section we show, perhaps surprisingly, that a simple greedy algorithm relying on carefully defined cost functions to evaluate the greedy moves guarantees a competitive ratio matching this lower bound and is therefore optimal.

In the remainder of the paper, instead of working directly with cost functions c_r, it will be often convenient to work with their rescaled version, referred to in the literature as latency functions. Specifically, the *latency function* ℓ_r associated with c_r is defined by $\ell_r := c_r(x)/x$ for any $x \in \mathbb{N}$, where we set $\ell_r(0) := 0$ for convenience. Here, ℓ_r can be interpreted as the cost of resource r per task, assuming such a cost is equally shared among the tasks assigned to it. Within this setting, the standard greedy algorithms process tasks sequentially and irrevocably, and assign each task $t \in [T]$ to the action a_t minimizing the overall latency $\sum_{r\in\mathcal{R}:r\in a_t}\ell_r(|(a_1,\dots,a_{t-1})|_r)$ of task t, e.g., [19], or the total cost $\sum_{r\in\mathcal{R}:r\in a_t}c_r(|(a_1,\dots,a_{t-1})|_r)$, e.g., [4]. The greedy algorithms we consider in this work are instead more general than the previous ones and work as follows. Let LT be a *latency transformation*, that is, a map that associates the original latency function $\ell_r : \mathbb{N} \to \mathbb{R}_{\geq 0}$ of resource r with a (possibly) modified one $\hat{\ell}_r : \mathbb{N} \to \mathbb{R}_{\geq 0}$. In this work, we consider a *greedy algorithm with respect to modified latency functions derived from* LT. Specifically, we consider a greedy algorithm employing latency functions $\hat{\ell}_r := LT(\ell_r)$ in place of the original latency functions ℓ_r, and assigning each task t to the action a_t minimizing the overall modified latency $\sum_{r\in a_t}\hat{\ell}_r(|(a_1,\dots,a_{t-1})|_r)$ of task t.[3]

A pseudo-code of the greedy algorithm with modified latency functions is given in Algorithm 1.

[3] We assume that ties are resolved according to an arbitrary deterministic tie-braking rule.

Algorithm 1 Greedy Algorithm with Modified Latency Functions.

Input: Instance I, latency transformation LT
Initialize: $\hat{\ell}_r := LT(\ell_r)$ for all $r \in \mathcal{R}$ ▷ Compute modified latency functions
for $t = 1, \dots, T$ **do**
  Choose $a_t \in \arg\min_{a \in \mathcal{A}_t} \sum_{r \in \mathcal{R}: r \in a_t} \hat{\ell}_r(|(a_1, \dots, a_t)|_r)$ ▷ Greedy with respect to $\hat{\ell}_r$
end for
return $a = (a_1, \dots, a_T)$

As the main result of this section, we show that the greedy algorithm with modified latency obtained from the solution of a specific linear program (LP) provides a competitive ratio that matches the lower bound provided in Theorem 1, when the considered cost functions satisfy a very mild convexity requirement. In the following we discuss our three-step approach to achieve this goal, and then state the main result in Theorem 7.

First, for a given cost function c, consider the following infinite LP in the variables $(\lambda, (\hat{\ell}(y))_{y \in \mathbb{N}})$, denoted as LP($c$):

$$\inf_{\lambda, (\hat{\ell}(y))_{y \in \mathbb{N}}} \quad \lambda$$
$$\text{s.t.} \quad \lambda \cdot c(x) \geq c(y) - \sum_{j=1}^{y} \hat{\ell}(j) + x \cdot \hat{\ell}(y+1) \qquad \forall x, y \in \mathbb{N}_0$$
$$\hat{\ell}(y+1) \geq \hat{\ell}(y), \quad \hat{\ell}(y) \geq 0 \qquad \forall y \in \mathbb{N} \qquad \text{LP(c)}$$

Equipped with this linear program, Lemma 4 establishes that the greedy algorithm employing the latency transformation LT has a competitive ratio no higher than λ when the input instance has cost functions in $\mathcal{C}$, so long as λ is simultaneously feasible for all the above LPs, i.e., for LP(c), for all $c \in \mathcal{C}$. As becomes evident in the proof of Lemma 4, LP(c) arises from imposing the greedy choice on each task, compared to any other choice.

Second, under mild convexity assumptions on the resource costs, Lemma 6 shows that, by appropriately choosing a specific latency transformation LT^*, the value $\lambda^* = \sup_{c \in \mathcal{C}} \mathsf{CR}(c)$ simultaneously satisfies all LPs. Third, and final, by putting Lemma 4 and 6 together, Theorem 7 shows that the competitive ratio of the greedy algorithm associated with LT^* is at most $\lambda^* = \sup_{c \in \mathcal{C}} \mathsf{CR}(c)$. This implies that $\mathsf{CR}(c)$ is a tight bound on the optimal competitive ratio and the algorithm we have designed is optimal.

▶ **Lemma 4.** *Let $\lambda \geq 0$ and LT be a latency transformation such that $(\lambda, \hat{\ell})$, with $\hat{\ell} = LT(\ell)$ and $\ell(x) = c(x)/x$, is feasible for LP(c), for all $c \in \mathcal{C}$. Then, the competitive ratio of the greedy algorithm over instances generated by $\mathcal{C}$ with modified latency functions $\hat{\ell}$ is at most λ.*

Proof. Let $I = (\mathcal{R}, \{c_r\}_{r \in \mathcal{R}}, \{\mathcal{A}_t\}_{t=1}^T)$ be a given input instance with cost functions in $\mathcal{C}$, such that each pair $(\lambda, \hat{\ell}_r)$ with $\hat{\ell}_r = LT(\ell_r)$ is feasible for any $r \in \mathcal{R}$. Let $a^{\text{GR}} = (a_1^{\text{GR}}, \dots, a_T^{\text{GR}})$ and $\tilde{a} = (\tilde{a}_1, \dots, \tilde{a}_T)$ be the allocation of the greedy algorithm with modified latency functions $\hat{\ell}_r = LT(\ell_r)$ and any other allocation at the end of the time horizon, respectively. Denote with $y_r = |a^{\text{GR}}|_r$ and $x_r = |\tilde{a}|_r$ the congestion of resource $r \in \mathcal{R}$ in allocations a^{GR} and $\tilde{a}$, respectively.
As $\lambda \cdot c(x) \geq c(y) - \sum_{t=1}^{y} \hat{\ell}(t) + x \cdot \hat{\ell}(y+1)$ for all $x, y \in \mathbb{N}$, it specifically holds for x_r, y_r, for any resource r. Thus, summing over all resources, we obtain

$$\sum_{r \in \mathcal{R}} \lambda \cdot c(|\tilde{a}|_r) \geq \sum_{r \in \mathcal{R}} c(|a^{\text{GR}}|_r) - \sum_{r \in \mathcal{R}} \sum_{j=1}^{|a^{\text{GR}}|_r} \hat{\ell}(j) + \sum_{r \in \mathcal{R}} |\tilde{a}|_r \cdot \hat{\ell}(|a^{\text{GR}}|_r + 1).$$

As $\sum_{r\in\mathcal{R}} c(|\tilde{a}|_r) = C(\tilde{a})$ and $\sum_{r\in\mathcal{R}} c(|a^{\mathrm{GR}}|_r) = C(a^{\mathrm{GR}})$, showing that $\sum_{r\in\mathcal{R}} |\tilde{a}|_r \cdot \hat{\ell}(|a^{\mathrm{GR}}|_r + 1) - \sum_{r\in\mathcal{R}} \sum_{j=1}^{|a^{\mathrm{GR}}|_r} \hat{\ell}(j) \geq 0$ suffices to conclude, as then $\lambda \cdot C(\tilde{a}) \geq C(a^{\mathrm{GR}}) - \sum_{r\in\mathcal{R}} \sum_{j=1}^{|a^{\mathrm{GR}}|_r} \hat{\ell}(j) + \sum_{r\in\mathcal{R}} |\tilde{a}|_r \cdot \hat{\ell}(|a^{\mathrm{GR}}|_r + 1) \geq C(a^{\mathrm{GR}})$ and $\lambda \geq \frac{C(a^{\mathrm{GR}})}{C(\tilde{a})}$ follows.

Let $|a^{\mathrm{GR}}|_{r,t} = |(a_1^{\mathrm{GR}}, \dots, a_t^{\mathrm{GR}})|_r \coloneqq |\{j \in [t] : r \in a_j^{\mathrm{GR}}\}|$ denote the congestion of resource r up to and including time step t, when employing the greedy algorithm with modified latency functions $\hat{\ell}_r$. Observe that the action a_t^{GR} according to the greedy algorithm at time step t minimizes the sum of the modified latency functions of the corresponding resources up to and including time step t. Thus, $a_t^{\mathrm{GR}} \in \arg\min_{a\in\mathcal{A}_t} \sum_{r\in\mathcal{R}:r\in a_t} \hat{\ell}_r(|a^{\mathrm{GR}}|_{r,t-1} + 1)$ and, for any other action $\tilde{a}_t$, it holds that $\sum_{r\in\mathcal{R}:r\in\tilde{a}_t} \hat{\ell}(|a^{\mathrm{GR}}|_{r,t-1} + 1) \geq \sum_{r\in\mathcal{R}:r\in a_t^{\mathrm{GR}}} \hat{\ell}(|a^{\mathrm{GR}}|_{r,t-1} + 1)$.

Aggregating $0 \leq \sum_{r\in\mathcal{R}:r\in\tilde{a}_t} \hat{\ell}(|a^{\mathrm{GR}}|_{r,t-1} + 1) - \sum_{r\in\mathcal{R}:r\in a_t^{\mathrm{GR}}} \hat{\ell}(|a^{\mathrm{GR}}|_{r,t-1} + 1)$ over all time steps, we obtain

$$
\begin{aligned}
0 &\leq \sum_{t=1}^{T} \left(\sum_{r\in\mathcal{R}:r\in\tilde{a}_t} \hat{\ell}(|a^{\mathrm{GR}}|_{r,t-1} + 1) - \sum_{r\in\mathcal{R}:r\in a_t^{\mathrm{GR}}} \hat{\ell}(|a^{\mathrm{GR}}|_{r,t-1} + 1) \right) \\
&= \sum_{t=1}^{T} \left(\sum_{r\in\mathcal{R}:r\in\tilde{a}_t} \hat{\ell}(|a^{\mathrm{GR}}|_{r,t-1} + 1) - \sum_{r\in\mathcal{R}:r\in a_t^{\mathrm{GR}}} \hat{\ell}(|a^{\mathrm{GR}}|_{r,t}) \right) \\
&\leq \sum_{t=1}^{T} \left(\sum_{r\in\mathcal{R}:r\in\tilde{a}_t} \hat{\ell}(|a^{\mathrm{GR}}|_{r} + 1) - \sum_{r\in\mathcal{R}:r\in a_t^{\mathrm{GR}}} \hat{\ell}(|a^{\mathrm{GR}}|_{r,t}) \right) \\
&= \sum_{r\in\mathcal{R}} |\tilde{a}|_r \hat{\ell}(|a^{\mathrm{GR}}|_r + 1) - \sum_{r\in\mathcal{R}} \sum_{j=1}^{|a^{\mathrm{GR}}|_r} \hat{\ell}(j)
\end{aligned}
$$

and the second inequality follows as $\hat{\ell}(x+1) \geq \hat{\ell}(x)$ for all $x \in \mathbb{N}$ and $|a^{\mathrm{GR}}|_r \geq |a^{\mathrm{GR}}|_{r,t}$ for all $t \in [T]$. ◀

▶ Remark 5. We observe that the optimal value of each LP(c) is at least 1. Specifically, by leveraging the condition $\hat{\ell}(2) \geq \hat{\ell}(1)$ in the constraint corresponding to $(x, y) = (1, 1)$, we obtain $\mathrm{LP}(c) \geq 1$. Furthermore, if c is a non-negative linear function of the form $c(x) = \alpha x$, the optimal value of $\mathrm{LP}(c)$ is $\lambda = 1$, which can be trivially achieved by setting $\hat{\ell}(x) = \alpha$ for all x. Thus, in light of Lemma 4, linear functions with non-negative coefficients can be regarded as *dummy functions* that do not influence the analysis of (the upper bound on) the competitive ratio. Consequently, they can be excluded from the considered class $\mathcal{C}$ without loss of generality.

From this point onward, the results we obtain hold under very mild convexity assumptions on the considered cost functions. Specifically, we consider *strictly convex latency functions*, meaning they satisfy condition $c(h+1) - c(h) > c(h) - c(h-1)$ for any $h \in \mathbb{N}$. We note that this requirement is broad enough to encompass most functions of interest, including polynomials, exponentials, polylogarithms, and many others. In fact, these functions are either strictly convex or linear, and the latter type can be discarded from our analysis without loss of generality (by Remark 5).

▶ **Lemma 6.** *Let $\mathcal{C}$ be a class of strictly convex cost functions. Then, if $\sup_{c\in\mathcal{C}} \mathsf{CR}(c) < \infty$, there exists a latency transformation LT^* such that $(\lambda^*, \hat{\ell}^*)$, with $\lambda^* = \sup_{c\in\mathcal{C}} \mathsf{CR}(c)$, $\hat{\ell}^* = LT^*(\ell)$, and $\ell(x) = c(x)/x$, is feasible for LP(c), for all $c \in \mathcal{C}$.*

Proof. Given $c \in \mathcal{C}$, let $\lambda^*(c) \coloneqq \mathsf{CR}(c) < \infty$. To show the claim, it will be sufficient to show that $\lambda^*(c)$ is feasible for LP(c), for any $c \in \mathcal{C}$. Indeed, by setting $\lambda^* := \sup_{c\in\mathcal{C}} \lambda^*(c)$, the claim of the lemma follows. From this point onward, the proof of the lemma can be divided into two main parts, which we outline below:

Part 1: Consider the following linear program with infinite constraints and variables $(\hat{L}(y))_{y\in\mathbb{N}_0}$, derived from the linear program LP(c) by setting $\hat{L}(y) \coloneqq \sum_{j=1}^{y} \hat{\ell}(j)$ for any $y \in \mathbb{N}_0$, removing the monotonicity constraints and imposing $\hat{L}(y) \geq c(y)$ for any $y \in \mathbb{N}$:

$$\begin{aligned} \inf_{\lambda,(\hat{L}(y))_{y\in\mathbb{N}_0}} \quad & \lambda \\ \text{s.t. } & \lambda \cdot c(x) \geq c(y) - \hat{L}(y) + x \cdot (\hat{L}(y+1) - \hat{L}(y)) \qquad \forall x \in \mathbb{N}, y \in \mathbb{N}_0 \\ & \hat{L}(0) = 0,\ \hat{L}(y) \geq c(y) \quad \forall y \in \mathbb{N}. \end{aligned} \tag{LP2(c)}$$

Let $(\hat{L}^*(y))_{y\in\mathbb{N}_0}$ be the sequence of values iteratively defined by $\hat{L}^*(0) \coloneqq 0$ and

$$\hat{L}^*(y+1) \coloneqq \min_{x\in\mathbb{N}} \frac{\lambda^*(c) \cdot c(x) - c(y) + (x+1) \cdot \hat{L}^*(y)}{x} \quad \forall y \in \mathbb{N}_0, \tag{3}$$

where the above minimum is well-defined as the right-hand part of (3), which can be written as $\lambda^*(c)\frac{c(x)}{x} + \hat{L}^*(y) + \frac{L^*(y)-c(y)}{x}$, goes to infinite for $x \to \infty$, due to the strict convexity of c and $\lambda^*(c) = \mathsf{CR}(c) \geq 1$. By exploiting the iterative definition of $\hat{L}^*(y)$ for all $y \in \mathbb{N}$ and the definition of $\lambda^*(c) = \mathsf{CR}(c)$, we show that $(\lambda^*(c), \hat{L}^*)$ is a feasible solution for LP2(c).

Part 2: Let $(\lambda^*(c), \hat{\ell}^*)$ be the pair obtained from the previous solution of LP2(c), by setting $\hat{\ell}^*(y+1) \coloneqq \hat{L}^*(y+1) - \hat{L}^*(y)$ for any $y \in \mathbb{N}_0$. By exploiting the feasibility of $(\lambda^*(c), \hat{L}^*)$ for LP2(c), we show that $(\lambda^*(c), \hat{\ell}^*)$ is also a feasible solution for the linear program LP(c).

Proof of Part 1 of Lemma 6. Given $x, y \in \mathbb{N}_0$, let $const_{\hat{L}}(x, y)$ denote the constraint associated with (x, y) in LP2(c). We will first show that the solution $(\lambda^*(c), \hat{L}^*)$, with $\hat{L}^*$ determined iteratively from (3), is feasible for LP2(c). We observe that each constraint $const_{\hat{L}}(x, y)$ can be written as $\frac{\lambda \cdot c(x) - c(y) + (x+1)\cdot \hat{L}(y)}{x} \geq \hat{L}(y+1)$. Thus, the solution $(\lambda^*, \hat{L}^*)$ provided in (3) satisfies all constraints $const_{\hat{L}}(x, y)$ with $x \in \mathbb{N}$ and $y \in \mathbb{N}_0$.

It remains to show that $\hat{L}^*(y) \geq c(y)$ holds for any $y \in \mathbb{N}$. As outlined in the observation below (3), for any $y \in \mathbb{N}_0$ there exists $x_y \in \mathbb{N}$ that minimizes the right-hand side of (3), that is,

$$\hat{L}^*(y+1) = \frac{\lambda^*(c) \cdot c(x_y) - c(y) + (x_y + 1) \cdot \hat{L}^*(y)}{x_y}. \tag{4}$$

By using (4), one can show by induction that

$$\hat{L}^*(y+1) = \lambda^*(c) \left(\sum_{i=0}^{y} \frac{c(x_i)}{x_i} \prod_{h=i+1}^{y} \frac{x_h + 1}{x_h} \right) - \left(\sum_{i=0}^{y} \frac{c(i)}{x_i} \prod_{h=i+1}^{y} \frac{x_h + 1}{x_h} \right), \quad \forall y \in \mathbb{N}_0 \tag{5}$$

(see Lemma 11 in the appendix for a proof).

Thus, for any $y \in \mathbb{N}_0$, we get

$$\begin{aligned} \frac{\hat{L}^*(y+1) - c(y+1)}{\sum_{i=0}^{y} \frac{c(x_i)}{x_i} \prod_{h=i+1}^{y} \frac{x_h+1}{x_h}} &= \lambda^*(c) - \frac{\sum_{i=0}^{y} \frac{c(i)}{x_i} \prod_{h=i+1}^{y} \frac{x_h+1}{x_h} + c(y+1)}{\sum_{i=0}^{y} \frac{c(x_i)}{x_i} \prod_{h=i+1}^{y} \frac{x_h+1}{x_h}} \\ &= \mathsf{CR}(c) - \frac{\sum_{i=0}^{y} \frac{c(i)}{x_i} \prod_{h=1}^{i} \frac{x_h}{x_h+1} + c(y+1) \prod_{h=1}^{y} \frac{x_h}{x_h+1}}{\sum_{i=0}^{y} \frac{c(x_i)}{x_i} \prod_{h=1}^{i} \frac{x_h}{x_h+1}} \geq 0, \end{aligned}$$

where the first equality holds by (5), the second equality is obtained by using $\lambda^*(c) = \mathsf{CR}(c)$ and dividing both the numerator and denominator of the fraction by $\prod_{h=1}^{y} \frac{x_h}{x_h+1}$, and the last inequality follows from the definition of $\mathsf{CR}(c)$ (see (1)). The above inequalities imply $\hat{L}^*(y+1) - c(y+1) \geq 0$ for any $y \in \mathbb{N}_0$, that is, the constraint $\hat{L}^*(y) - c(y) \geq 0$ of LP2(c) is satisfied for any $y \in \mathbb{N}$. This shows the feasibility of $(\lambda^*(c), \hat{L}^*)$ for LP2(c), and concludes the proof of Part 1.

Proof of Part 2 of Lemma 6. Let $(\lambda^*(c), \hat{\ell}^*)$ be the pair obtained by setting $\hat{\ell}^*(y+1) := \hat{L}^*(y+1) - \hat{L}^*(y)$ for any $y \in \mathbb{N}$, that is, $L^*(y+1) = \sum_{j=1}^{y+1} \hat{\ell}^*(j)$. Given $x, y \in \mathbb{N}_0$, let $const_{\hat{\ell}}(x, y)$ denote the constraint in LP(c) associated with (x, y). By the results obtained in the proof of Part 1, we have that all constraints $const_{\hat{\ell}}(x, y)$ with $x \neq 0$ are satisfied. Thus, to prove the feasibility of $(\lambda^*(c), \hat{\ell}^*)$ for LP(c), it remains to show that $const_{\hat{\ell}}(0, y)$ is satisfied for any $y \in \mathbb{N}_0$ and that the monotonicity constraints $\hat{\ell}(y+1) \geq \hat{\ell}(y)$ hold.

We observe that $\sum_{j=1}^{y} \hat{\ell}^*(j) = \hat{L}^*(y) \geq c(y)$ holds for any $y \in \mathbb{N}$, where the inequality follows from the feasibility of the last constraint of LP2(c) (shown in Part 1). As the obtained inequality is equivalent to $const_{\hat{\ell}}(0, y)$ for $(\lambda, \hat{\ell}) = (\lambda^*(c), \hat{\ell}^*)$, we have that $const_{\hat{\ell}}(0, y)$ is satisfied by $(\lambda^*(c), \hat{\ell}^*)$.

Now, we show that the monotonicity constraints $\hat{\ell}^*(y+1) \geq \hat{\ell}^*(y)$ hold. Assume by contradiction that there exists $y' \in \mathbb{N}_0$ such that $\hat{\ell}^*(y'+1) < \ell^*(y')$. Let $(x_y)_{y \in \mathbb{N}}$ denote the sequence of integers defined in the proof of Part 1, that is, the sequence satisfying (4). By the convexity of cost function c, the tightness of constraints $const_{\hat{L}}(x_y, y)$ (by (4)) and the feasibility of $const_{\hat{L}}(x, y)$ for any $x \in \mathbb{N}$, one can show that $\hat{\ell}^*$ is decreasing in $y \geq y'$. The proof of this property is deferred to Lemma 12 in Section B, and is shown by induction on $y \geq y'$.

Then, by exploiting the strict convexity of c and the decreasing monotonicity of $\ell^*(y)$ in $y \geq y'$ we have that, under solution $(\lambda^*(c), \hat{\ell}^*)$, the right-hand side of $const_{\hat{\ell}}(1, y)$ tends to ∞ for y tending to ∞ (see Lemma 13 in Section B). However, this violates the feasibility of $(\lambda^*(c), \hat{\ell}^*)$ for a constraint $const_{\hat{\ell}}(1, y)$ with a sufficiently large y, thus yielding a contradiction.

Thus, $\ell^*(y)$ is necessarily non-decreasing in y, which concludes the proof of Part 2, and therefore the entire proof of Lemma 6. ◀

By Lemma 6, we conclude that there exists a latency transformation LT^* such that the resulting modified latency functions $\hat{\ell}^*$, associated with cost functions in $\mathcal{C}$, ensure the feasibility of $\lambda^* = \sup_{c \in \mathcal{C}} \mathsf{CR}(c)$ in the linear program LP(c) for all $c \in \mathcal{C}$. Then, by Lemma 4, we obtain the following main result of the section:

▶ **Theorem 7.** *Let $\mathcal{C}$ be a class of strictly convex cost functions. Then, there exists a latency transformation LT^* such that $\sup_{c \in \mathcal{C}} \mathsf{CR}(c)$ is an upper bound on the competitive ratio of the greedy algorithm with modified latencies derived from LT^*, over instances with cost functions drawn from $\mathcal{C}$.*

5 Computing modified cost functions

While the result in Theorem 7 is existential, its proof suggests that optimal modified cost functions can be computed through the solution of LP(c). However, this is an *infinite* linear program so that it is unclear how such modified cost functions should be computed, and whether their computation can be carried out efficiently. This will constitute the focus of this section, which we divide in two parts. First, we will consider the case where an upper bound on the total congestion on each resource is known (Section 5.1). We will then move beyond this case in Section 5.2.

5.1 Upper bound on congestion

If an upper bound $\bar{m}$ on the total congestion at any resource is known, for example because we are aware of the maximum number of tasks appearing, it is possible to limit the number of constraints appearing in LP(c) to finitely many, specifically by considering constraints for $x, y \leq \bar{m}$ only. This intuition is made formal in the ensuing corollary.

▶ **Corollary 8.** *Consider the class of input instances obtained employing cost functions in $\mathcal{C}$, and such that the total congestion on any resource is at most $\bar{m} \in \mathbb{N}$. Let, for given $c \in \mathcal{C}$ and $\bar{m}$, $\bar{\lambda}(c, \bar{m})$ be the optimal value of the linear program LP(c,$\bar{m}$), and let $\lambda^\star \coloneqq \sup_{c\in\mathcal{C}} \bar{\lambda}(c, \bar{m})$. Then we can compute, in polynomial time, a latency transformation LT such that the greedy algorithm with modified latency functions derived from LT achieves a competitive ratio no greater than $\lambda^\star$. Furthermore, if $\mathcal{C}$ is made of strictly convex functions, the resulting competitive ratio is no greater than $\sup_{c\in\mathcal{C}} \mathsf{CR}(c)$.*

$$\begin{aligned} \inf_{\lambda,\ (\hat{\ell}(y))_{y\in[\bar{m}+1]}} \quad & \lambda \\ s.t. \quad & \lambda \cdot c(x) \geq c(y) - \sum_{j=1}^{y} \hat{\ell}(j) + x \cdot \hat{\ell}(y+1) \quad \forall x, y \in [\bar{m}]_0 \\ & \hat{\ell}(y+1) \geq \hat{\ell}(y) \geq 0 \qquad\qquad\qquad\qquad \forall y \in [\bar{m}] \end{aligned} \tag{LP(c,$\bar{m}$)}$$

Proof. Observe that, for given $c \in \mathcal{C}$ and $\bar{m} \in \mathbb{N}$, LP(c,$\bar{m}$) is feasible, as, for arbitrary $\hat{\ell}$ satisfying $\hat{\ell}(y+1) \geq \hat{\ell}(y) \geq 0$ and $\sum_{j=1}^{y} \hat{\ell}(j) \geq c(y)$ for all $y \in [\bar{m}]$ (the latter constraint equals the top constraint in LP(c,$\bar{m}$) for $x = 0$), we can find λ large enough such that the top constraint in LP(c,$\bar{m}$) is satisfied for all $x, y \in [\bar{m}]_0$.
Let $(\bar{\lambda}(c, \bar{m}), \ell^\star)$ be an optimal solution to LP(c,$\bar{m}$).[4] Observe that $(\bar{\lambda}(c, \bar{m}), \ell^\star)$ is computable in polynomial time since it is the solution of a finite linear program. Denote $\lambda^\star \coloneqq \sup_{c\in\mathcal{C}} \bar{\lambda}(c, \bar{m})$. As $\lambda^\star \cdot c(x) \geq \bar{\lambda}(c, \bar{m}) \cdot c(x)$ and $(\bar{\lambda}(c, \bar{m}), \ell^\star)$ is feasible for LP$(c, \bar{m})$, so is $(\lambda^\star, \ell^\star)$. Let a^{GR} be the allocation of the greedy algorithm with modified latency functions $\ell^\star$, and let $\tilde{a}$ be any other allocation. By definition of the class of input instances, for any resource r, $|a^{\mathrm{GR}}|_r, |\tilde{a}|_r \leq \bar{m}$. Thus, by considering the above restriction for each resource congestion in the proof of Lemma 4, we can immediately obtain that, given λ that is simultaneously feasible for all LP$(c, \bar{m})$, λ is an upper bound on the competitive ratio - in other words, we obtain an equivalent statement of Lemma 4, but adapted to this case of bounded resource congestion. As we have already shown that $(\lambda^\star, \ell^\star)$ is feasible for each LP$(c, \bar{m})$, it follows that $\lambda^\star$ is an upper bound on the competitive ratio.
If $\mathcal{C}$ contains strictly convex functions only, suppose $\sup_{c\in\mathcal{C}} \mathsf{CR}(c) < \infty$, otherwise the second statement is trivial. From the proof of Lemma 6, we know that $\lambda = \sup_{c\in\mathcal{C}} \mathsf{CR}(c)$ is simultaneously feasible for each LP(c). Since $\lambda^\star$ is the optimal value for LP(c,$\bar{m}$) for at least one $c \in \mathcal{C}$ and the feasible region of LP(c,$\bar{m}$) contains that of LP(c), any value λ that is simultaneously feasible for each LP(c) must also be feasible for each LP$(c, \bar{m})$. Therefore, $\sup_{c\in\mathcal{C}} \mathsf{CR}(c) \geq \lambda^\star$, and consequently, $\sup_{c\in\mathcal{C}} \mathsf{CR}(c)$ is, just like $\lambda^\star$, an upper bound on the competitive ratio. ◀

A very well-studied class of problems is that where latency functions are polynomials with non-negative coefficients and degree no higher than d, i.e., extracted from the set $\{\sum_{k=0}^{d} \alpha_k x^k, \alpha_k \geq 0\}$. Note that, in these settings, it suffices to compute a modified latency for each individual monomial x^k, $k = 0, \ldots, d$ as explained in general terms in the following remark. This allows to compute and store ahead of time the modified latency functions.

[4] The dependence of $\ell^\star$ on c and $\bar{m}$ is for sake of readability omitted.

▶ Remark 9. When the latency functions used to generate the instance are obtained from the set $\{\sum_{k=0}^{d} \alpha_k b_k(x), \alpha_k \geq 0\}$, i.e., they are linear combination of some given basis functions $b_0, \dots, b_d$ taking the form $\ell_r(x) = \sum_{k=0}^{d} \alpha_{k,r} b_k(x)$, then the modified latencies $\hat{\ell}_r(x) = \sum_{k=0}^{d} \alpha_{k,r} \hat{b}_k(x)$, obtained by linearly combining $\hat{b}_k = LT^*(b_k)$, as per Theorem 7, also achieve optimal competitive ratio. For details, we refer to Section C.

Notice that the transformation defined by LP(c,$\bar{m}$) achieves a competitive ratio that depends on $\bar{m}$ and may be strictly better than $\sup_{c \in \mathcal{C}} \mathsf{CR}(c)$, converging to the latter as $\bar{m}$ grows. Table 2 in Section D illustrates how such value increases with $\bar{m}$ for polynomial latency functions with non-negative coefficients of varying maximum degree d.

5.2 No upper bound on congestion

Without an explicit upper bound on the total congestion or time horizon, it is unclear whether modified cost functions can be efficiently computed as the linear program LP(c) contains infinitely many constraints. To tackle this issue, we take the following approach. The idea is to fix some finite $\bar{m}$, compute the modified resource costs up until $\bar{m}$, by accounting for finitely many constraints only, and then extend the solution from $\bar{m}+1$ to infinity with a carefully defined function. For this purpose, it is convenient to work with the sum of the modified latency functions, defined as $\hat{L}(y) = \sum_{j=1}^{y} \hat{\ell}(j)$, where we let y range in $1, \dots, \bar{m}+2$, instead of working directly with $\hat{\ell}(y)$. Note that, given $\hat{L}(y)$, one can reconstruct $\hat{\ell}(y+1) = \hat{L}(y+1) - \hat{L}(y)$ and similarly, given $\hat{\ell}$, one can compute $\hat{L}$ using the above definition. We investigate this approach for polynomial latency functions $\ell_r(x) = \sum_{k=0}^{d} \alpha_{k,r} x^k$ with non-negative coefficients. For ease of readability, we present the approach for monomial latency functions x^k. Using Remark 9, we easily obtain modified latency functions for general polynomials $\ell_r(x) = \sum_{k=0}^{d} \alpha_{k,r} x^k$ with identical competitive ratio. Further, by Remark 5, we can disregard the constant monomial x^0 (which has cost function $c(x) = x$), as $\hat{\ell}(y) = 1$ (likewise, $\hat{L}(y) = y$) achieves competitive ratio equal to 1. Therefore, w.l.o.g., we consider latency functions $\{x^1, \dots, x^d\}$, and for each given monomial x^k consider the following (finite) linear program

$$
\begin{aligned}
(\lambda_k^\star, \hat{L}_k^\star) \in \arg \inf_{\lambda,\ (\hat{L}(y))_{y \in [\bar{m}+2]_0}} \quad & \lambda \\
\text{s.t.} \quad & \lambda x^{k+1} \geq y^{k+1} - \hat{L}(y) + x \cdot (\hat{L}(y+1) - \hat{L}(y)) \qquad \forall x, y \in [\bar{m}]_0 \\
& \hat{L}(y+1) - \hat{L}(y) \geq \hat{L}(y) - \hat{L}(y-1) \qquad \forall y \in [1, \dots, \bar{m}+1] \\
& \hat{L}(0) = 0, \quad \hat{L}(y) = \beta y^{k+1} \qquad \forall y \in \{\bar{m}+1, \bar{m}+2\},\ \beta = k+1. \qquad (6)
\end{aligned}
$$

Here, the first set of constraints represents the constraints already appearing in LP(c), the second set of constraints ensures non-decreasingness of $\hat{\ell}$, while the third set imposes that the sum of the modified latencies $\hat{L}(y)$ matches the monomial βy^{k+1} at the cut-off point $y = \bar{m}+1$. This choice will be crucial as the monomial βy^{k+1} will be used to patch $\hat{L}$ for $y = \bar{m}+1$ and beyond. Armed with this, we can state the main result of this section, which allows us to evaluate the competitive ratio of the resulting patched modified latency functions. Due to space limitations, we refer to the full version of the paper for the proof.

▶ **Theorem 10.** *Consider the class of instances generated by polynomial latencies of degree no higher than $d \in \mathbb{N}$, i.e., where $\ell_r(y) = \sum_{k=0}^{d} \alpha_{k,r} y^k$. Consider modified latency functions obtained as $\hat{\ell}_r(y) = \sum_{k=0}^{d} \alpha_{k,r} \hat{b}_k(y)$, where $\hat{b}_k(y+1) = \hat{L}_k(y+1) - \hat{L}_k(y)$ and*

$$
\hat{L}_k(y) = \begin{cases} \hat{L}_k^\star(y) & \text{for } y \in [\bar{m}]_0 \\ \beta \cdot y^{d+1} & \text{for } y \geq \bar{m}+1 \end{cases}, \qquad \text{where} \qquad \beta = k+1. \qquad (7)
$$

Then, the greedy algorithm employing such modified latencies achieves a competitive ratio no higher than

$$\max_{k\in[d]}\left\{\lambda_k^\star, \left(\frac{(\bar{m}+2)^{d+1}-(\bar{m}+1)^{d+1}}{(\bar{m}+1)^d}\right)^{d+1}\right\}.$$

By Lemma 14, $\lim_{\bar{m}\to\infty}\left(\frac{(\bar{m}+2)^{d+1}-(\bar{m}+1)^{d+1}}{(\bar{m}+1)^d}\right)^{d+1} = (d+1)^{d+1}$, hence the second term recovers the lower bound on the competitive ratio obtained in Corollary 3 in the limit.

The competitive ratio above is presented with the explicit choice $\beta = d+1$ for computing and patching $\hat{L}$. In principle, (7) can be computed for arbitrary $\beta > 1$, as long as the same β is used in (6) as in the patched $\hat{L}_k$. In that case we obtain, for fixed $\beta > 1$, the bound $\lambda_k(\beta) = \max\{\lambda_k^\star(\beta), \lambda_k^\Delta(\beta)\}$, and

$$\lambda_k^\Delta(\beta) = \left(\frac{(\bar{m}+2)^{k+1}-(\bar{m}+1)^{k+1}}{(\bar{m}+1)^k}\right)^{k+1} \beta\left(\frac{\beta}{\beta-1}\right)^k\left[\frac{k^k}{(k+1)^{k+1}}\right]. \tag{8}$$

Although optimizing $\lambda_k(\beta)$ over β may allow for slightly better competitive ratio, it is unclear how to do so *efficiently* and whether there exists an explicit expression of the optimal β. If $\bar{m}$ is small, $\lambda_k^\Delta(\beta)$ outweighs $\lambda_k^\star(\beta)$, and our choice $\beta = (d+1)$ is motivated by achieving the lowest possible $\lambda_k^\Delta(\beta)$ of all $\beta > 1$, which is especially relevant for small $\bar{m}$, whilst at the same time yielding good values for $\lambda_k(\beta)$ compared to other β's for larger $\bar{m}$ (see Table 1 and Table 3 for $k = 1$ and $k = 2$, respectively). We leave the question of optimal choice of β for future research.

Table 1 Comparison of $\lambda_1(\beta) = \max\{\lambda_1^\star(\beta), \lambda_1^\Delta(\beta)\}$ for $\ell(x) = x$, for varying β and $\bar{m}$.

	$\beta = d+0.5$		$\beta = d+0.9$		$\beta = d+1$		$\beta = d+1.1$		$\beta = d+1.5$	
$\bar{m}$	$\lambda_1^\star$	λ_1^Δ	$\lambda_1^\star$	λ_1^Δ	$\lambda_1^\star$	λ_1^Δ	$\lambda_1^\star$	λ_1^Δ	$\lambda_1^\star$	λ_1^Δ
1	2.33	7.03	2.87	6.27	3	**6.25**	3.13	6.26	3.67	6.51
2	2.79	6.13	3.30	5.46	3.43	**5.44**	3.56	5.46	4.07	5.67
5	3.29	5.28	3.73	4.71	3.85	**4.69**	3.97	4.71	4.43	4.89
10	3.55	4.92	3.91	4.38	4.01	**4.37**	4.10	4.38	4.51	4.55
20	3.71	4.72	3.99	4.20	4.07	**4.19**	4.15	4.20	4.49	4.37
40	3.81	4.61	4.02	4.11	4.08	**4.10**	4.15	4.11	4.44	4.27
100	3.88	4.54	**4.03**	4.05	4.08	4.04	4.13	4.05	4.38	4.21
200	3.92	4.52	**4.03**	4.03	4.07	4.02	4.11	4.03	4.33	4.19
400	3.94	4.51	**4.03**	4.02	4.06	4.01	4.10	4.02	4.30	4.18

$\star$) For given β and $\bar{m}$, $\lambda_1^\star(\beta)$ equals the optimal value of (6) with the respective value of β, whereas $\lambda_1^\Delta(\beta)$ equals the value in (8). Both values are rounded to two decimal places. In bold the minimal value of $\lambda_1(\beta) = \max\{\lambda_1^\star(\beta), \lambda_1^\Delta(\beta)\}$ among the reported β-values.

6 Beyond Online Optimization - A Game-Theoretic Perspective

Standard *Congestion Games* [39] constitute the game-theoretic framework most closely related to our model. In these games, tasks $t \in [T]$ act as rational and selfish agents, each autonomously choosing its action a_t to minimize its *individual cost* $\sum_{r\in a_t}\ell(|a|_r)$, expressed by means of the latency functions $\ell_r(x) = c_r(x)/x$. The impact of selfish behavior is

often quantified via the *Price of Anarchy* [30], which, similarly to the competitive ratio, represents the worst-case ratio between the total cost of stable solutions arising from agents' interactions (i.e., Nash equilibria [37]) and the optimal cost. Significant bulk of work addresses the problem of designing taxes or incentives to reduce as much as possible the price of anarchy [7, 12, 17, 18, 20, 25, 24, 28, 38]. Similarly as in our online optimization model, [11] provide taxation mechanisms for agents that act myopically and greedily select the action that minimize their individual costs. The optimality of the resulting greedy outcomes is measured through the competitive analysis adopted for our model. However, their taxation mechanism uses the full knowledge of the instance and is therefore unusable in online settings. In this context, we observe that the modified latency functions provided by our general greedy algorithm can be exploited to define taxes or incentives that guide agents toward better outcomes. Specifically, if we charge each agent selecting a resource r an amount $\delta_r(x) := \hat{\ell}_r(x) - \ell_r(x)$, with x being the congestion of resource r after the agent's selection, and $\delta_r(x)$ representing a tax if positive and an incentive if negative, the greedy choices of the agents "simulate" an execution of the greedy algorithm with modified latencies $(\hat{\ell}_r)_{r\in\mathcal{R}}$. Indeed, the individual cost of the agent processed at time step $t \in [T]$, when choosing action a_t, becomes $\sum_{r\in a_t}(\delta_r(|(a_1,\dots,a_t)|_r) + \ell_r(|(a_1,\dots,a_t)|_r)) = \sum_{r\in a_t}\hat{\ell}_r(|(a_1,\dots,a_t)|_r)$, that is, the cost used by our algorithm to perform the greedy assignment.

We conclude that the decentralized intervention mechanism, in which each resource autonomously calculates a tax/incentive $\delta_r(x)$ to charge/credit the agents who select it, guarantees the same competitive ratio analyzed in our work on the online optimization front. Furthermore, the optimality of our greedy framework among all online algorithms translates into the optimality of the resulting intervention mechanisms among all others.

References

1 Susanne Albers. Energy-efficient algorithms. *Communications of the ACM*, 53(5):86–96, 2010. doi:10.1145/1735223.1735245.

2 Matthew Andrews, Antonio Fernández Anta, Lisa Zhang, and Wenbo Zhao. Routing for energy minimization in the speed scaling model. In *2010 Proceedings IEEE INFOCOM*, pages 1–9. IEEE, 2010. doi:10.1109/INFCOM.2010.5462071.

3 James Aspnes, Yossi Azar, Amos Fiat, Serge Plotkin, and Orli Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *Journal of the ACM (JACM)*, 44(3):486–504, 1997. doi:10.1145/258128.258201.

4 Baruch Awerbuch, Yossi Azar, Edward F. Grove, Ming-Yang Kao, P. Krishnan, and Jeffrey Scott Vitter. Load balancing in the l_p norm. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 383–391, 1995. doi:10.1109/SFCS.1995.492494.

5 Yossi Azar, Andrei Z. Broder, and Anna R. Karlin. On-line load balancing. *Theor. Comput. Sci.*, 130(1):73–84, 1994. doi:10.1016/0304-3975(94)90153-8.

6 Yossi Azar, Joseph Naor, and Raphael Rom. The competitiveness of on-line assignments. *J. Algorithms*, 18(2):221–237, 1995. doi:10.1006/JAGM.1995.1008.

7 Martin J. Beckmann, C. B. McGuire, and Christopher B. Winsten. *Studies in the Economics of Transportation*. Yale University Press, 1956.

8 Vittorio Bilò. A unifying tool for bounding the quality of non-cooperative solutions in weighted congestion games. *Theory Comput. Syst.*, 62(5):1288–1317, 2018. doi:10.1007/S00224-017-9826-1.

9 Vittorio Bilò, Angelo Fanelli, Michele Flammini, and Luca Moscardelli. Performances of one-round walks in linear congestion games. *Theory of Computing Systems*, 49(1):24–45, 2011. doi:10.1007/S00224-010-9309-0.

10 Vittorio Bilò and Cosimo Vinci. On the impact of singleton strategies in congestion games. In *Proceedings of the 25th Annual European Symposium on Algorithms, (ESA)*, pages 17:1–17:14, 2017. doi:10.4230/LIPICS.ESA.2017.17.

11 Vittorio Bilò and Cosimo Vinci. Dynamic taxes for polynomial congestion games. *ACM Trans. Economics and Comput.*, 7(3):15:1–15:36, 2019. doi:10.1145/3355946.

12 Vittorio Bilò and Cosimo Vinci. *Coping with Selfishness in Congestion Games - Analysis and Design via LP Duality*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2023. doi:10.1007/978-3-031-30261-9.

13 Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

14 Niv Buchbinder and Joseph Naor. The design of competitive online algorithms via a primal-dual approach. *Found. Trends Theor. Comput. Sci.*, 3(2-3):93–263, 2009. doi:10.1561/0400000024.

15 Ioannis Caragiannis. Better bounds for online load balancing on unrelated machines. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 972–981, 2008. URL: http://dl.acm.org/citation.cfm?id=1347082.1347188.

16 Ioannis Caragiannis, Michele Flammini, Christos Kaklamanis, Panagiotis Kanellopoulos, and Luca Moscardelli. Tight bounds for selfish and greedy load balancing. *Algorithmica*, 61(3):606–637, 2011. doi:10.1007/S00453-010-9427-8.

17 Ioannis Caragiannis, Christos Kaklamanis, and Panagiotis Kanellopoulos. Improving the efficiency of load balancing games through taxes. In *Internet and Network Economics, 4th International Workshop, WINE 2008, Proceedings*, volume 5385, pages 374–385, 2008. doi:10.1007/978-3-540-92185-1_43.

18 Ioannis Caragiannis, Christos Kaklamanis, and Panagiotis Kanellopoulos. Taxes for linear atomic congestion games. *ACM Trans. Algorithms*, 7(1):13:1–13:31, 2010. doi:10.1145/1868237.1868251.

19 George Christodoulou, Vahab S. Mirrokni, and Anastasios Sidiropoulos. Convergence and approximation in potential games. *Theoretical Computer Science*, 438:13–27, 2012. doi:10.1016/J.TCS.2012.02.033.

20 Richard Cole, Yevgeniy Dodis, and Tim Roughgarden. How much can taxes help selfish routing? *Journal of Computer and System Sciences*, 72(3):444–467, 2006. doi:10.1016/J.JCSS.2005.09.010.

21 Yuval Emek, Shay Kutten, Ron Lavi, and Yangguang Shi. Approximating generalized network design under (dis)economies of scale with applications to energy efficiency. *J. ACM*, 67(1), 2020. doi:10.1145/3377387.

22 Babak Farzad, Neil Olver, and Adrian Vetta. A priority-based model of routing. *Chicago Journal of Theoretical Computer Science*, 2008(1), 2008. URL: http://cjtcs.cs.uchicago.edu/articles/2008/1/contents.html.

23 Amos Fiat and Gerhard J. Woeginger. *Online algorithms: The state of the art*, volume 1442. Springer, 1998.

24 Miriam Fischer, Martin Gairing, and Dario Paccagnan. Fair interventions in weighted congestion games. In *International Conference on Web and Internet Economics*. Springer, 2024. To appear.

25 Martin Gairing and Dario Paccagnan. In congestion games, taxes achieve optimal approximation. *Operations Research*, 72(3):966–982, 2023. doi:10.1287/OPRE.2021.0526.

26 Tobias Harks, Stefan Heinz, and Marc E. Pfetsch. Competitive online multicommodity routing. *Theory of Computing Systems*, 45(3):533–554, 2009. doi:10.1007/S00224-009-9187-5.

27 Anna R. Karlin, Mark S. Manasse, Larry Rudolph, and Daniel D. Sleator. Competitive snoopy caching. *Algorithmica*, 3:79–119, 1988.

28 Jon M. Kleinberg and Sigal Oren. Mechanisms for (mis)allocating scientific credit. *Algorithmica*, 84(2):344–378, 2022. doi:10.1007/S00453-021-00902-Y.

29 Max Klimm, Daniel Schmand, and Andreas Tönnis. The online best reply algorithm for resource allocation problems. In *Proceedings of the 12th International Symposium on Algorithmic Game Theory (SAGT)*, pages 200–215, 2019. doi:10.1007/978-3-030-30473-7_14.

30 Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th Annual Conference on Theoretical Aspects of Computer Science (STACS)*, pages 404–413, 1999. doi:10.1007/3-540-49116-3_38.

31 Stefano Leonardi. On-line network routing. *Online Algorithms: The State of the Art*, pages 242–267, 2005.

32 Thomas Lücking, Marios Mavronicolas, Burkhard Monien, and Manuel Rode. A new model for selfish routing. *Theoretical Computer Science*, 406(3):187–206, 2008. doi:10.1016/J.TCS.2008.06.045.

33 Konstantin Makarychev and Maxim Sviridenko. Solving optimization problems with diseconomies of scale via decoupling. *J. ACM*, 65(6):42:1–42:27, 2018. doi:10.1145/3266140.

34 H. Brendan McMahan. A survey of algorithms and analysis for adaptive online learning. *Journal of Machine Learning Research*, 18(90):1–50, 2017. URL: https://jmlr.org/papers/v18/14-428.html.

35 Carol A. Meyers and Andreas S. Schulz. The complexity of welfare maximization in congestion games. *Networks*, 59(2):252–260, 2012. doi:10.1002/NET.20439.

36 Viswanath Nagarajan and Lily Wang. Online generalized network design under (dis)economies of scale. *Mathematics of Operations Research*, 49(1):107–124, 2024. doi:10.1287/MOOR.2022.1349.

37 John F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Science*, 36(1):48–49, 1950.

38 Dario Paccagnan, Rahul Chandan, and Jason R. Marden. Utility and mechanism design in multi-agent systems: An overview. *Annu. Rev. Control.*, 53:315–328, 2022. doi:10.1016/J.ARCONTROL.2022.02.002.

39 Robert W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

40 Tim Roughgarden. Barriers to near-optimal equilibria. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 71–80. IEEE, 2014. doi:10.1109/FOCS.2014.16.

41 Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2012. doi:10.1561/2200000018.

42 Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985. doi:10.1145/2786.2793.

43 Subhash Suri, Csaba D. Tóth, and Yunhong Zhou. Selfish load balancing and atomic congestion games. *Algorithmica*, 47(1):79–96, 2007. doi:10.1007/S00453-006-1211-4.

44 Cosimo Vinci. Non-atomic one-round walks in congestion games. *Theoretical Computer Science*, 764:61–79, 2019. doi:10.1016/J.TCS.2018.06.038.

A Missing Proof of Corollary 3 from Section 3

By Theorem 1, it is sufficient to show that $\mathsf{CR}(c) \geq (d+1)^{d+1}$, for c being the monomial function $c(x) = x^{d+1}$ with $d \geq 0$. Let $(x_i)_{i \in \mathbb{N}_0}$ be an infinite sequence of natural numbers defined by $x_i := \lceil i/(d+1) \rceil$ for any $i \in \mathbb{N}_0$. By definition of $\mathsf{CR}(c)$, we have

$$\mathsf{CR}(c) \geq \lim_{y\to\infty} \frac{\sum_{i=0}^{y} \frac{c(i)}{x_i} \prod_{h=1}^{i} \frac{x_h}{x_h+1} + c(y+1)\prod_{h=1}^{y} \frac{x_h}{x_h+1}}{\sum_{i=0}^{y} \frac{c(x_i)}{x_i} \prod_{h=1}^{i} \frac{x_h}{x_h+1}} \tag{9}$$

$$= \lim_{y\to\infty} \frac{\sum_{i=0}^{y} \frac{i^{d+1}}{\lceil i/(d+1)\rceil} \prod_{h=1}^{i} \frac{\lceil h/(d+1)\rceil}{\lceil h/(d+1)\rceil+1} + (y+1)^{d+1}\prod_{h=1}^{y} \frac{\lceil h/(d+1)\rceil}{\lceil h/(d+1)\rceil+1}}{\sum_{i=0}^{y} \frac{\lceil i/(d+1)\rceil^{d+1}}{\lceil i/(d+1)\rceil} \prod_{h=1}^{i} \frac{\lceil h/(d+1)\rceil}{\lceil h/(d+1)\rceil+1}} \tag{10}$$

$$\geq \lim_{y\to\infty} \frac{\sum_{i=0}^{y} \frac{i^{d+1}}{\lceil i/(d+1)\rceil} \prod_{h=1}^{i} \frac{\lceil h/(d+1)\rceil}{\lceil h/(d+1)\rceil+1}}{\sum_{i=0}^{y} \frac{\lceil i/(d+1)\rceil^{d+1}}{\lceil i/(d+1)\rceil} \prod_{h=1}^{i} \frac{\lceil h/(d+1)\rceil}{\lceil h/(d+1)\rceil+1}} \tag{11}$$

$$= \lim_{y\to\infty} \frac{\sum_{i=0}^{y} \frac{i^{d+1}}{\lceil i/(d+1)\rceil} \left(\frac{1}{\lceil i/(d+1)\rceil+1}\right)^{1+(i-1)\%(d+1)} \left(\frac{1}{\lceil i/(d+1)\rceil}\right)^{d-(i-1)\%(d+1)}}{\sum_{i=0}^{y} \frac{\lceil i/(d+1)\rceil^{d+1}}{\lceil i/(d+1)\rceil} \left(\frac{1}{\lceil i/(d+1)\rceil+1}\right)^{1+(i-1)\%(d+1)} \left(\frac{1}{\lceil i/(d+1)\rceil}\right)^{d-(i-1)\%(d+1)}} \tag{12}$$

$$= (d+1)^{d+1} \lim_{y\to\infty} \frac{\sum_{i=0}^{y} \frac{(i/(d+1))^{d+1}}{\lceil i/(d+1)\rceil} \left(\frac{1}{\lceil i/(d+1)\rceil+1}\right)^{1+(i-1)\%(d+1)} \left(\frac{1}{\lceil i/(d+1)\rceil}\right)^{d-(i-1)\%(d+1)}}{\sum_{i=0}^{y} \frac{\lceil i/(d+1)\rceil^{d+1}}{\lceil i/(d+1)\rceil} \left(\frac{1}{\lceil i/(d+1)\rceil+1}\right)^{1+(i-1)\%(d+1)} \left(\frac{1}{\lceil i/(d+1)\rceil}\right)^{d-(i-1)\%(d+1)}} \tag{13}$$

$$= (d+1)^{d+1} \lim_{y\to\infty} \frac{(d+1)H(\lceil y/(d+1)\rceil)}{(d+1)H(\lceil y/(d+1)\rceil)} \tag{14}$$

$$= (d+1)^{d+1}, \tag{15}$$

where (9) holds by definition of $\mathsf{CR}(c)$, (12) holds by exploiting the telescoping properties of product $\prod_{h=1}^{i} \frac{\lceil h/(d+1)\rceil}{\lceil h/(d+1)\rceil+1}$ (with $a\%b$ denoting the remainder of a divided by b) and (14) holds since both the numerator and the denominator of (13) are asymptotically equal to $(d+1)H(\lceil y/(d+1)\rceil)$, with $H(n) = \sum_{i=1}^{n} \frac{1}{n}$ denoting the n-th harmonic number.

B Missing Results and Proofs from Section 4

▶ **Lemma 11.**

$$\hat{L}^*(y+1) = \lambda^*(c)\left(\sum_{i=0}^{y} \frac{c(x_i)}{x_i} \prod_{h=i+1}^{y} \frac{x_h+1}{x_h}\right) - \left(\sum_{i=0}^{y} \frac{c(i)}{x_i} \prod_{h=i+1}^{y} \frac{x_h+1}{x_h}\right), \quad \forall y \in \mathbb{N}_0.$$

Proof. We prove the statement with induction over y.
Base case ($y = 0$): We have

$$\hat{L}^*(1) = \frac{\lambda^*(c)\cdot c(x_0) - c(0) + (x_0+1)\cdot \hat{L}^*(0)}{x_0,} = \frac{\lambda^*(c)\cdot c(x_0) - c(0)}{x_0},$$

where we use $\hat{L}^*(0) = 0$. Further, we have

$$\lambda^*(c)\left(\sum_{i=0}^{0} \frac{c(x_i)}{x_i} \prod_{h=i+1}^{0} \frac{x_h+1}{x_h}\right) - \left(\sum_{i=0}^{0} \frac{c(i)}{x_i} \prod_{h=i+1}^{0} \frac{x_h+1}{x_h}\right) = \lambda^*(c)\frac{c(x_0)}{x_0} - \frac{c(0)}{x_0},$$

where we use $\prod_{h=1}^{0} \frac{x_h+1}{x_h} = 1$.
Inductive step: Assume the statement holds for $y = n$. Hence,

$$\hat{L}^*(n+1) = \frac{\lambda^*(c)\cdot c(x_n) - c(n) + (x_n+1)\cdot \hat{L}^*(n)}{x_n}$$

$$= \lambda^*(c)\left(\sum_{i=0}^{n} \frac{c(x_i)}{x_i} \prod_{h=i+1}^{n} \frac{x_h+1}{x_h}\right) - \left(\sum_{i=0}^{n} \frac{c(i)}{x_i} \prod_{h=i+1}^{n} \frac{x_h+1}{x_h}\right)$$

Then, for $y = n+1$,

$$\begin{aligned}
\hat{L}^*(n+2) &= \frac{\lambda^*(c)\cdot c(x_{n+1}) - c(n+1) + (x_{n+1}+1)\cdot \hat{L}^*(n+1)}{x_{n+1}} \\
&= \frac{\lambda^*(c)\cdot c(x_{n+1}) - c(n+1)}{x_{n+1}} \\
&\quad + \frac{(x_{n+1}+1)}{x_{n+1}}\left[\lambda^*(c)\cdot\left(\sum_{i=0}^{n}\frac{c(x_i)}{x_i}\prod_{h=i+1}^{n}\frac{x_h+1}{x_h}\right) - \left(\sum_{i=0}^{n}\frac{c(i)}{x_i}\prod_{h=i+1}^{n}\frac{x_h+1}{x_h}\right)\right] \\
&= \frac{\lambda^*(c)\cdot c(x_{n+1}) - c(n+1)}{x_{n+1}} + \lambda^*(c)\cdot\left(\sum_{i=0}^{n}\frac{c(x_i)}{x_i}\prod_{h=i+1}^{n+1}\frac{x_h+1}{x_h}\right) \\
&\quad - \left(\sum_{i=0}^{n}\frac{c(i)}{x_i}\prod_{h=i+1}^{n+1}\frac{x_h+1}{x_h}\right) \\
&= \prod_{h=n+2}^{n+1}\frac{x_h+1}{x_h}\left(\frac{\lambda^*(c)\cdot c(x_{n+1}) - c(n+1)}{x_{n+1}}\right) \\
&\quad + \lambda^*(c)\cdot\left(\sum_{i=0}^{n}\frac{c(x_i)}{x_i}\prod_{h=i+1}^{n+1}\frac{x_h+1}{x_h}\right) - \left(\sum_{i=0}^{n}\frac{c(i)}{x_i}\prod_{h=i+1}^{n+1}\frac{x_h+1}{x_h}\right) \\
&= \lambda^*(c)\cdot\left(\sum_{i=0}^{n+1}\frac{c(x_i)}{x_i}\prod_{h=i+1}^{n+1}\frac{x_h+1}{x_h}\right) - \left(\sum_{i=0}^{n+1}\frac{c(i)}{x_i}\prod_{h=i+1}^{n+1}\frac{x_h+1}{x_h}\right),
\end{aligned}$$

where we use $\prod_{h=n+2}^{n+1}\frac{x_h+1}{x_h} = 1$. ◀

▶ **Lemma 12.** *If $\hat{\ell}^*(y'+1) < \hat{\ell}^*(y')$ holds for some $y' \in \mathbb{N}$, then $\hat{\ell}^*$ is decreasing in integers $y \geq y'$.*

Proof. We show, by induction on $y \geq y'$, that $\hat{\ell}^*(y+1) < \hat{\ell}^*(y)$ holds for any integer $y \geq y'$.

The base case $y = y'$ trivially holds by the assumptions. Assume that the claim holds for $y = n$, and let us show it for $y = n+1$. By exploiting the feasibility of $const_{\hat{L}}(x, y)$ for any $x, y \in \mathbb{N}$ (shown in Part 1 of Lemma 6) and the tightness of $const_{\hat{L}}(x, y)$ for $x = x_n$ and any $y \in \mathbb{N}$ (arising from (4)), we proceed as follows. When considering solution $(\lambda^*(c), \hat{L}^*)$ and using $\hat{L}^*(y) = \sum_{j=1}^{y}\hat{\ell}^*(j)$, constraint $const_{\hat{L}}(x_n, n)$ can be written as $\lambda^*(c)\cdot c(x_n) = c(n) - \sum_{j=1}^{n}\hat{\ell}^*(j) + x_n\cdot\hat{\ell}^*(n+1)$, and $const_{\hat{L}}(x_n, n-1)$ can be written as $\lambda^*(c)\cdot c(x_n) \geq c(n-1) - \sum_{j=1}^{n-1}\hat{\ell}^*(j) + x_n\cdot\hat{\ell}^*(n)$. Since the left-hand sides of $const_{\hat{L}}(x_n, n)$ and $const_{\hat{L}}(x_n, n-1)$ are equal, these constraints imply that $c(n) - \sum_{j=1}^{n}\hat{\ell}^*(j) + x_n\cdot\hat{\ell}^*(n+1) \geq c(n-1) - \sum_{j=1}^{n-1}\hat{\ell}^*(j) + x_n\cdot\hat{\ell}^*(n)$, that is:

$$c(n) - c(n-1) - \hat{\ell}^*(n) + x_n(\hat{\ell}^*(n+1) - \hat{\ell}^*(n)) \geq 0. \tag{16}$$

Similarly, by exploiting the equality constraint $const_{\hat{L}}(x_n, n)$ and the inequality constraint $const_{\hat{L}}(x_n, n+1)$, we obtain

$$c(n+1) - c(n) - \hat{\ell}^*(n+1) + x_n(\hat{\ell}^*(n+2) - \hat{\ell}^*(n+1)) \leq 0. \tag{17}$$

Combining (16) and (17) gives

$$\begin{aligned}
&x_n[\hat{\ell}^*(n+2) - \hat{\ell}^*(n+1)] - x_n[\hat{\ell}^*(n+1) - \hat{\ell}^*(n)] \\
&\qquad \leq [c(n) - c(n-1) - \hat{\ell}^*(n)] - [c(n+1) - c(n)) - \hat{\ell}^*(n+1)] < 0
\end{aligned} \tag{18}$$

where the last inequality holds since $c(n+2)-c(n+1) \geq c(n+1)-c(n)$ (by the convexity of cost function c) and $\hat{\ell}^*(n+1)-\ell^*(n) < 0$ (by the inductive hypothesis). By (18) and $\hat{\ell}^*(n+1)-\hat{\ell}^*(n) < 0$ we have $\hat{\ell}^*(n+2)-\hat{\ell}^*(n+1) < \hat{\ell}^*(n+1)-\hat{\ell}^*(n) < 0$, and this shows the inductive step. ◀

▶ **Lemma 13.** *Let $f : \mathbb{N} \to \mathbb{R}$ be the function defined as $f(y) := c(y) - \sum_{j=1}^{y} \hat{\ell}^*(j) + \hat{\ell}^*(y+1)$. If $\hat{\ell}^*$ is decreasing in $y \geq y'$, then $\lim_{y\to\infty} f(y) = \infty$.*

Proof. Given $y \geq y'$, we have $f(y) = c(y) - \sum_{j=1}^{y} \hat{\ell}^*(j) + \hat{\ell}^*(y+1) \geq c(y) - \sum_{j=1}^{y} \hat{\ell}^*(j) \geq c(y) - \sum_{j=1}^{y'} \hat{\ell}^*(j) - (y-y') \cdot \hat{\ell}^*(y')$, where the last inequality holds by the non-decreasing monotonicity of $\hat{\ell}^*$ on $y \geq y'$. By the monotonicity and strict convexity of cost function c, we necessarily have that $\lim_{y\to\infty}(c(y) - \sum_{j=1}^{y'} \hat{\ell}^*(j) - (y-y') \cdot \hat{\ell}^*(y')) = \infty$. Thus, by the previous inequalities, $\lim_{y\to\infty} f(y) = \infty$ must also hold. ◀

C Missing Proofs from Section 5

Further details on Remark 9.

▶ Remark. When the latency functions used to generate the instance are obtained from the set $\{\sum_{k=0}^{d} \alpha_k b_k(x), \alpha_k \geq 0\}$, i.e., they are linear combination of some given basis functions $b_0, \ldots, b_d$ taking the form $\ell_r(x) = \sum_{k=0}^{d} \alpha_{k,r} b_k(x)$, then the modified latencies $\hat{\ell}_r(x) = \sum_{k=0}^{d} \alpha_{k,r} \hat{b}_k(x)$, obtained by linearly combining $\hat{b}_k = LT^*(b_k)$, as per Theorem 7, also achieve optimal competitive ratio.

Proof. To see this, denote with $\overline{\mathsf{CR}} = \max_k \mathsf{CR}(b_k)$ the largest competitive ratio over all individual basis b_k, and notice that the optimal competitive ratio $\sup_{c\in\mathcal{C}} \mathsf{CR}(c)$ cannot be lower than $\overline{\mathsf{CR}}$. To see that the modified latencies in fact attain a competitive ratio equal to $\overline{\mathsf{CR}}$, and therefore are optimal, one can simply observe that $\overline{\mathsf{CR}}$ must be feasible for the linear program to evaluate the performance of a given generic latency of the form $\ell_r(x) = \sum_{k=0}^{d} \alpha_{k,r} b_k(x)$. In more detail, as, for each basis function b_k, $(\mathsf{CR}(b_k), \hat{b}_k)$ is feasible for the linear program LP(c) corresponding to b_k (i.e., LP(c) with $c_r(x) = \sum_{k=0}^{d} \alpha_{k,r} b_k(x) \cdot x$), and $\overline{\mathsf{CR}} \cdot b_k(x) \geq \mathsf{CR}(b_k) \cdot b_k(x)$ for any $x \in \mathbb{N}_0$, $(\overline{\mathsf{CR}}, \hat{b}_k)$ is also feasible for LP(c). Thus, as for every basis function b_k, $(\overline{\mathsf{CR}}, \hat{b}_k)$ satisfies all constraints of LP(c), $\overline{\mathsf{CR}}$ taken with any linear, non-negative combination of modified basis functions $\hat{\ell}_r(x) = \sum_{k=0}^{d} \alpha_{k,r} \hat{b}_k(x)$ then also is feasible for the *same* combination of basis functions $\ell_r(x) = \sum_{k=0}^{d} \alpha_{k,r} b_k(x)$. ◀

▶ **Lemma 14.** *For fixed $d > 0$,*

$$\lim_{y\to\infty} \left(\frac{(y+1)^{d+1} - y^{d+1}}{y^d} \right)^{d+1} = (d+1)^{d+1}.$$

Proof. The proof employs the binomial approximation $(1+x)^n = 1 + nx + \mathcal{O}(x^{-2})$ for x, nx sufficiently small.
Consider the term $(1+\frac{1}{y})^{d+1}$. For sufficiently large y such that $(d+1)\cdot(1/y) \ll 1$, $\left(1+\frac{1}{y}\right)^{d+1} = 1 + (d+1)\frac{1}{y} + \mathcal{O}(y^{-2})$. This yields, for sufficiently large y,

$$\begin{aligned}\left(\frac{(y+1)^{d+1} - y^{d+1}}{y^d} \right)^{d+1} &= \left(y \left(\frac{y+1}{y} \right)^{d+1} - y \right)^{d+1} \\ &= \left(y \left(1 + \frac{1}{y} \right)^{d+1} - y \right)^{d+1}\end{aligned}$$

$$= \left(y \left(1 + (d+1)\frac{1}{y} + \mathcal{O}(y^{-2}) \right) - y \right)^{d+1}$$
$$= \left(y + (d+1) + \mathcal{O}(y^{-1}) - y \right)^{d+1}$$
$$= \left((d+1) + \mathcal{O}(y^{-1}) \right)^{d+1} .inutes$$

As $y \to \infty$, $\mathcal{O}(y^{-1}) \to 0$, which leaves the constant term $(d+1)^{d+1}$. ◀

D Other Results, Tables and Graphs

▶ Remark 15. Let c be a non-null, non-negative function with some $y', x' \in \mathbb{N}$ such that $c(y') > 0$ and $c(x') = 0$. Then, the competitive ratio of any online algorithm over instances employing c is unbounded.

Proof. Let $y' \coloneqq arg\,min_{t>0}\{c(t) | c(t) > 0\}$. By the assumption on c, $y' > 0$, thus $c(y') > 0$. Further, $c(x') = 0$, and $c(y'+1) \geq 0$ as c is non-negative. Hence, setting $y \coloneqq y'$ and $x_h \coloneqq x'$ for any $h \in [y]_0$ in (1), the numerator is strictly positive, whereas the denominator is zero, thus $\mathsf{CR}(c) = \infty$. Then, by exploiting the lower bound instance provided in Theorem 1 with these values of y and $(x_h)_{h \in [y]_0}$, we have that no online algorithm can achieve a competitive ratio over instances generated by c below ∞. ◀

Table 2 Achievable competitive ratios for polynomial latency functions and bounded congestion $\bar{m}$, for varying $\bar{m}$ (rounded to two decimal places).

	Maximum degree d of latency function		
$\bar{m}$	1	2	3
1	1	1	1
2	2	4	8
5	2.88	10.75	46.38
10	3.30	15.63	91.60
20	3.56	19.33	137.55
40	3.71	21.81	174.93
100	3.82	23.76	205.82
200	3.88	24.64	219.97
400	3.91	25.25	238.68

Table 3 Comparison of $\lambda_2(\beta) = \max\{\lambda_2^\star(\beta), \lambda_2^\Delta\}$ for $\ell(x) = x^2$ for varying β and $\bar{m}$.

	$\beta = d + 0.5$		$\beta = d + 0.9$		$\beta = d + 1$		$\beta = d + 1.1$		$\beta = d + 1.5$	
$\bar{m}$	$\lambda_2^\star(\beta)$	λ_2^Δ	$\lambda_2^\star(\beta)$	λ_2^Δ	$\lambda_2^\star(\beta)$	λ_2^Δ	$\lambda_2^\star(\beta)$	λ_2^Δ	$\lambda_2^\star(\beta)$	λ_2^Δ
1	7	110.26	8.07	107.27	8.33	**107.17**	8.60	107.26	9.67	108.92
2	11.07	71.48	12.61	69.54	13	**69.48**	13.39	69.54	14.93	70.62
5	17.29	45.17	19.26	43.94	19.75	**43.90**	20.25	43.94	22.22	44.62
10	21.29	36.34	23.21	35.35	23.71	**35.32**	24.20	35.35	26.18	35.90
20	23.76	32.01	25.46	31.14	25.90	**31.11**	26.34	31.14	28.11	31.62
40	25.07	29.88	26.47	29.07	26.83	**29.04**	27.20	29.06	28.71	29.51
100	25.90	28.61	26.94	27.84	27.23	**27.81**	27.52	27.83	28.73	28.27
200	26.21	28.20	27.05	27.43	27.29	**27.41**	27.53	27.43	28.57	27.85
400	26.41	27.99	27.19	**27.23**	27.37	27.20	27.55	27.22	28.54	27.65

$\star$) For given β and $\bar{m}$, $\lambda_2^\star(\beta)$ equals the optimal value of (6) with the respective value of β, whereas $\lambda_2^\Delta(\beta)$ equals the value in (8). Both values are rounded to two decimal places. In bold the minimal value of $\lambda_2(\beta) = \max\{\lambda_2^\star(\beta), \lambda_2^\Delta(\beta)\}$ among the reported β-values.