

A New Approach to Capacity Scaling Augmented With Unreliable Machine Learning Predictions

Daan Rutten* and Debankur Mukherjee

Georgia Institute of Technology

January 29, 2021

Abstract

Modern data centers suffer from immense power consumption. The erratic behavior of internet traffic forces data centers to maintain excess capacity in the form of idle servers in case the workload suddenly increases. As an idle server still consumes a significant fraction of the peak energy, major data center operators like Amazon AWS have heavily invested in capacity scaling solutions. In simple terms, capacity scaling solutions aim to deactivate servers if the demand is low and to activate them again when the workload increases. To do so, an algorithm needs to strike a delicate balance between the power consumption, flow-time, and the switching cost. Over the last decade, the research community has developed sophisticated competitive online algorithms with worst-case guarantees. In the presence of historic data patterns, prescription from Machine Learning (ML) predictions typically outperform such competitive algorithms. This, however, comes at the cost of sacrificing the robustness of performance, since unpredictable surges in the workload are not uncommon. The current work builds on the emerging paradigm of augmenting unreliable ML predictions with online algorithms to develop novel robust algorithms that enjoy the benefits of both worlds.

In this paper, we analyze a continuous-time model for capacity scaling, where the goal is to minimize the weighted sum of flow-time, switching cost, and power consumption in an online fashion. The model generalizes much of the earlier related approaches. We propose a low-complexity algorithm, called the Adaptive Balanced Capacity Scaling (ABCS) algorithm, that has access to black-box ML predictions. Although ABCS is completely oblivious to the accuracy of these predictions, its performance does depend on the error of the predictions. In particular, if the predictions turn out to be accurate in hindsight, we prove that ABCS is $(1 + \epsilon)$ -competitive. Moreover, even when the predictions are inaccurate, ABCS guarantees a uniformly bounded competitive ratio. Finally, we investigate the performance of the ABCS algorithm on a real-world dataset and carry out extensive numerical experiments, which positively support the theoretical results.

Keywords — energy efficiency, online algorithms, competitive analysis, speed scaling, competitive ratio

*Email: drutten@gatech.edu

1 Introduction

1.1 Background and motivation

Modern data centers suffer from immense power consumption, which amounts to a massive economic and environmental impact. In 2014, data centers alone contributed to about 1.8% of the total U.S. electricity consumption [52] and this is projected to reach 7% in 2030 [46]. Consequently, data center providers are constantly striving to optimize their servers for energy efficiency, pushing the hardware’s efficiency to nearly its limit. At this point, algorithmic improvements appear to be critical in order to achieve substantial further gain [52]. A common practice for data centers has been to reserve significant excess service capacity in the form of idle servers [53], even though a typical idle server still consumes about 44% of its peak power consumption [52]. The recommendation from the U.S. Department of Energy [52], industry [17, 25, 47], and the academic research community [1, 21, 34] is, therefore, to implement dynamic capacity scaling functionality based on the demand. If the demand is low, the service capacity should be scaled down by deactivating servers, while at peak times, it should be scaled up by increasing the number of active servers. Instead of physically turning servers on or off, such dynamic scaling functionality is often implemented by carefully allocating a fraction of servers to other, lower priority services and quickly bringing them back at times of high demand; see [13, 51, 54] for a more detailed account. This maximizes the utilization of the system and hence minimizes the power consumption.

The call for algorithmic solutions to capacity scaling has inspired a vibrant line of research over the last decade [1, 5, 8, 19, 21, 27, 34, 35, 44, 45]. The problem fits into the framework of online algorithms, where the goal is to design algorithms that dynamically scale the current service capacity, based on the past and current system information. Here, the performance of an algorithm is captured in terms of the *Competitive Ratio* (CR), which is defined as the worst possible ratio between the cost incurred by the online algorithm and that by the offline optimum algorithm. Note that the online algorithm has information only about the past and the present, while the offline optimum has accurate information about all future input variables such as the task-arrival process in the context of the current article. The key advantage of such strong performance guarantees lies in its robustness, that is, the algorithm safeguards against the worst-case scenario.

However, any of today’s modern large-scale systems has access to massive historical data, which, combined with standard Machine Learning (ML) algorithms, can reveal definitive patterns. In these cases, simply following the recommendations obtained from the ML predictions typically outperforms any competitive algorithm. Netflix is an example of a company implementing capacity scaling in practice. Instead of relying on competitive online algorithms, Netflix has implemented ML algorithms in their Stryer system [47]. They noted that their demand usually follows regular patterns, allowing them to accurately predict the demand during a day based on data from previous weeks. Most of the time, the performance of the machine learning algorithm is therefore excellent. However, besides empirical verification, the performance of such ML predictions is not guaranteed. In fact, repeated observations show that unexpected surges in the workload are not at all uncommon [11, 32, 47], which cause a significant adverse impact on the system performance.

The contrasting approaches between academia and industry reveal a gap between what we are able to prove and what is desirable in practice. While online algorithms do not require any information about future arrivals, in practice, these predictions are usually available. At the

same time, an algorithm should not blindly trust the predictions because occasionally the accuracy of the predictions can be significantly poor. The current work aims to bridge this gap by incorporating ML predictions directly into the competitive analysis framework. In particular, we propose a novel low-complexity algorithm for capacity scaling, the Adaptive Balanced Capacity Scaling (ABCS) algorithm, which has access to a black-box predictor, lending predictions about future arrivals. Critically, not only is ABCS completely unaware of the prediction’s accuracy, we also restrain from making *any* statistical assumptions on the accuracy. Hence, this excludes any attempt to learn the prediction’s accuracy since accurate past predictions do not necessarily warrant the quality of future predictions. The main challenge therefore is to *design near-optimal algorithms which intelligently accept and reject the recommendations given by the ML predictor without knowing or learning their accuracy*. Note, however, that the performance of the ABCS algorithm does depend on the (unknown) error of the prediction, and it ensures, among others, two most desirable properties: (i) *consistency*, i.e., if the predictions turn out to be accurate in hindsight, then ABCS automatically replicates almost the optimal solution, and (ii) *competitiveness*, i.e., if the predictions are inaccurate in hindsight, then the performance of ABCS is at most an absolute constant factor times the minimum cost. The formal definitions of consistency and competitiveness are given in Section 3. It is worth emphasizing that this work is not concerned with how the ML predictions are obtained and uses them as a black box.

1.2 Our contributions

We will use a canonical continuous-time dynamical system model that is used to analyze algorithms for energy efficiency; see for example [1, 8, 34, 35, 37] for variations. Consider a system with a large number of homogeneous servers. Each server is in either of two states: *active* or *inactive*. Let $m(t)$ denote the number of active servers at time t . Workload arrives into the system in continuous time and gets processed at instantaneous rate $m(t)$. The system has a buffer of infinite capacity, where the unprocessed workload can wait until it is executed. We will assume that there is an unknown and arbitrary arrival rate function $\lambda(t)$ that represents the arrival process; see Section 2 for further details. We do not impose any restrictions on $\lambda(\cdot)$. To contrast this with the often-studied case when the workload arrival is stochastic, $\lambda(\cdot)$ can be thought of as an individual sample path of the corresponding stochastic arrival process. At any time, the system may decide to increase or decrease $m(t)$ in an online fashion. However, it pays a switching cost each time a server is activated. This represents the cost of terminating the lower priority service running at the inactive server and related migration costs [34, 35, 37, 51]. The goal of the system is to minimize the weighted sum of the flow-time, the switching cost, and the power consumption [37]. The flow-time is defined as the total time tasks spend in the system and is a measure of the response time [1, 8]. We will analyze the performance of an algorithm by its competitive ratio, the worst-case ratio between the cost of the online algorithm and the minimum offline cost, over all possible arrival rate functions $\lambda(\cdot)$. We further assume that the algorithm receives predictions about future workload through an ML oracle [36]. More precisely, at time $t = 0$, the ML oracle predicts an arrival rate function $\tilde{\lambda}(\cdot)$. The algorithm may use these predictions to increase or decrease the number of servers accordingly. For instance, if the oracle predicts that the demand in the next hour will increase, then the algorithm might proactively increase the number of servers. However, as mentioned before, it is crucial that the algorithm completely oblivious to the accuracy of these predictions. We measure the accuracy of the predictions in terms of the mean absolute error (MAE) between the predicted arrival rate function $\tilde{\lambda}$ and the actual rate function λ (see Definition 3.1). Our contributions in the current paper are twofold:

(1) Purely online algorithm with worst-case guarantees. First, we propose a novel purely online algorithm for capacity scaling, called Balanced Capacity Scaling (BCS). This purely online scenario, or the scenario of traditional competitive analysis, is equivalent to having predictions with infinite error. There are several fundamental works that have considered the purely online scenario for capacity scaling [19, 34, 35, 44, 45]. We extend the state of the art in this area by analyzing a more general model in continuous time and where unprocessed workload is allowed to wait. In fact, we show that a class of popular algorithms that were previously proposed are not constant competitive in the more general case (see Proposition 4.13). We show that the BCS algorithm is 5-competitive in the general case (Corollary 4.2) and is 2-competitive when waiting is not allowed and workload must be processed immediately upon arrival (Theorem 4.3). BCS is easy to implement and is memoryless, i.e., it only depends on the current state of the system and not on the past. Also, we prove a lower bound result that any deterministic online algorithm must have a competitive ratio of at least 2.549 (Proposition 4.4), which implies that the problem is strictly harder than the classical ski-rental problem, a benchmark for online algorithms.

(2) Augmenting unreliable ML predictions. When ML predictions are available, we first propose an adaptive algorithm, called Adapt to the Prediction (AP), which ensures consistency. That is, we prove (Theorem 4.6) that the competitive ratio of the AP algorithm is at most $1 + \Theta(\eta)$, where η is a suitable measure of the prediction's accuracy and is a function of the MAE between the predicted arrival rate function $\tilde{\lambda}$ and the actual rate function λ (Definition 3.2). The AP algorithm does not follow the predictions blindly. Rather, it dynamically scales the number of servers in an online fashion as the past predictions turn out to be inaccurate. Although the performance of the AP algorithm is near-optimal as $\eta \rightarrow 0$ and it degrades gracefully with η , it is not constant competitive if predictions are completely inaccurate ($\eta = \infty$). Thus, it does not provide any worst-case guarantees. This is a feat shared by many recent adaptive algorithms in the literature (see Remark 4.7).

Finally, we combine the ideas behind the BCS and the AP algorithms, to propose an algorithm that is both competitive *and* consistent. This brings us to the main contribution of the paper. We propose the Adaptive Balanced Capacity Scaling (ABCS) algorithm, which uses the structure of BCS and utilizes AP as a subroutine. ABCS has a hyperparameter $r \geq 1$, which can be fixed at any value before implementing the algorithm. It represents our confidence on the ML predictions. If we choose $r = 1$, then the algorithm works as a purely online one and disregards all predictions. In this case, ABCS is 5-competitive, as before. However, for any fixed $r > 1$, we prove (Corollary 4.9) that the competitive ratio of ABCS is at most

$$\min \left\{ (1 + \Theta(\eta)) (1 + \Theta(r^{-1})) , \Theta(r^3) \right\}, \quad (1.1)$$

where η is the prediction's accuracy as before. There are a number of consequences of the above result. We start by emphasizing that although the competitive ratio is a function of the error η , the algorithm is completely oblivious to it. Now, the higher we fix the value of r to be, the closer competitive ratio of ABCS gets to 1 if the predictions turn out to be accurate in hindsight. In this case, if the predictions are completely inaccurate ($\eta = \infty$), its competitive ratio is at most $\Theta(r^3)$, a constant that depends only on r and *not* on η . ABCS is therefore robust against unpredictable surges in workload, while providing near-optimal performance if the predictions are accurate.

Another interesting thing to note is that for $r > 1$, the competitive ratio in (1.1) is the minimum of two terms: the first term, which we call the *Optimistic Competitive Ratio* (OCR), is smaller when the prediction is accurate and the second term, which we call the *Pessimistic Competitive Ratio* (PCR), is smaller when the prediction is inaccurate. From the algorithm designer's perspec-

tive, there is a clear trade-off between OCR and PCR, which is conveniently controlled by the confidence hyperparameter r . It is important to note that ABCS provides performance guarantees for *any fixed* $r \geq 1$ irrespective of the model parameters or the accuracy of the predictions. However, *the choice of r reflects the risk that the system designer is willing to take in the pessimistic case against the gain in the optimistic case.* See Remark 4.10 for further discussion. This trade-off, however, is not specific to our algorithm. In fact, we prove a negative result in Proposition 4.12 that *any* algorithm which is $(1 + \delta)$ -competitive in the optimistic case, has a competitive ratio of at least $1/(4\delta^2)$ in the pessimistic case.

To test the performance of our algorithms in practice, we implemented them on both a real-world dataset of DNS requests observed at a campus network [39] and a set of artificial datasets, and the performance turns out to be excellent. See Section 5 for details.

1.3 Related works

Over the past two decades, the rapid growth of data centers and its immense power consumption have inspired a vibrant line of research in optimizing the energy efficiency of such systems [10, 15, 50, 55]. Below, we provide an overview of a few influential works relevant to the current paper.

The capacity scaling problem was introduced in a seminal paper by Lin et al. [34], who analyze a discrete time model of a data center. At each time step t , the cost of operating $m(t)$ servers is determined by the switching cost and an arbitrary convex function $g_t(m(t))$, which, for example, specifies the cost of increased power consumption versus response time. At time step t , the system reveals the function g_t and accurate functions $g_{t+1}, g_{t+2}, \dots, g_{t+w}$ in a prediction window of w future time steps. Lin et al. [34] propose an algorithm, called the LCP algorithm, and prove that it is 3-competitive. Surprisingly, the performance of the LCP algorithm does not improve if $w > 0$, i.e., if predictions are available. When g_t has a fixed form, our results generalize this work to continuous time, and to predictions that are not necessarily accurate. Moreover, the performance of our algorithm increases provably in the presence of predictions.

Lu et al. [35] consider a scenario where tasks cannot wait in queue and must be served immediately upon arrival. They discover that in this case, the capacity scaling problem reduces to solving a number of independent ski-rental problems. The authors then propose an algorithm and prove that it is 2-competitive. Our model, in addition, includes the response time, which directly generalizes the framework of [35]. This flexibility introduces a whole new dimension in the space of possible decisions. For example, since the results of Lu et al. [35] lack any form of delay, tasks are processed at the same time by any algorithm. Our model allows an algorithm-dependent delay of serving tasks, which *desynchronizes* the time at which tasks are processed at a server across different algorithms and hence, significantly complicates the analysis. Mazzucco and Dyachuk [40] analyze a related problem in which the number of servers is periodically updated and a task is lost if a server is not immediately available to serve it. The goal of their algorithm is to balance the power consumption and the cost of losing tasks. Galloway et al. [18] and later Gandhi et al. [20, 22] perform an empirical study of data centers. Their results show that significant power savings are possible, while maintaining much of the latency of the network.

A well-studied problem that is somewhat related to our setup is *speed scaling*. Here, the goal is to optimize the processing speed of a single server and to minimize the weighted sum of the flow-time and power consumption, while the switching cost is zero. In contrast to our model,

the scheduling of jobs also play a crucial role here. The power consumption is typically cubic in the processing speed. A seminal paper in this area is by Bansal et al. [8], which proposes an algorithm that schedules the task with the shortest remaining processing time (SRPT) first and processes it at a speed such that the power consumption is equal to the number of waiting tasks plus one. The authors prove that this algorithm is $(3 + \varepsilon)$ -competitive. Later papers have extended the case of the single server to processor sharing systems [56] and parallel processors with deadline constraints [2]. The problem of speed scaling has also been analyzed in the case the inter-arrival times and required processing times are exponentially distributed [4].

Any algorithm for the capacity scaling problem consists of two components: first, to activate servers and second, to deactivate servers. For a single server, a natural abstraction of the latter problem is the famous ski-rental problem, as first introduced by Karlin et al. [29]. The ski-rental problem has been applied to cases of capital investment [6, 14], TCP acknowledgement [28] and cache coherence [3]. Irani et al. [27] analyze the ski-rental problem when multiple power-down states are available, such as active, sleeping, hibernating, and inactive. The power consumption in each state is different and moving between the states incurs a switching cost. Augustine et al. [5] generalize these results when the transition costs between the different states are not additive. Although the current work focuses on only two states, i.e., active and inactive, we expect that the algorithm and proofs are general enough to accommodate multiple power-down states, which we leave as interesting future work. Khanafer et al. [30] analyze the ski-rental problem in a stochastic context.

Lykouris and Vassilvtskii [36] initiated the study of online algorithms augmented by ML predictions. They show how to adapt the marker algorithm for the caching problem to obtain a competitive ratio of 2 if the predictions are perfectly accurate, and a bounded competitive ratio if the predictions are inaccurate. The idea has since been applied to bipartite matching [31], ski-rental and scheduling on a single machine [48], bloom-filters [41], and frequency estimation [26]. Lee et al. [33] propose an algorithm which operates onsite generators to reduce the peak energy usage of data centers. Although related to the current work, their algorithm works independent to the capacity scaling happening inside the data center. Bamas et al. [7] discuss an algorithm augmented by predictions for the related problem of speed scaling discussed above, in the case of parallel processors with deadline constraints. Similar to our results, Bamas et al. [7] identify a trade-off between what we call an Optimistic Competitive Ratio and a Pessimistic Competitive Ratio. We prove, for the capacity scaling problem considered in the current work, that *any* algorithm must exhibit such a trade-off (see Proposition 4.12 for details). Mahdian et al. [38] propose an algorithm which naively switches between an optimistic and a pessimistic scheduling algorithm to minimize the makespan when routing tasks to multiple machines.

Recently, the notion of a predictor has also emerged in stochastic scheduling. Mitzenmacher [42] introduces the predictor as a probability density function $g(x, y)$ for a task with actual service time x and predicted service time y . Here the author analyzes the shortest predicted job first (SPJF) and shortest predicted remaining processing time (SPRPT) queueing disciplines for a single queue and determines the price of misprediction, i.e. the ratio of the cost if perfect information of the service time distribution is known and the cost if only predictions are available. For multiple queues, Mitzenmacher [43] has simulated the supermarket model or the ‘power-of- d ’ model, to show empirically that the availability of predictions greatly improves performance.

A different line of work called online algorithms with advice questions how many bits of *perfect* future information are necessary to reproduce the optimal offline algorithm (see [12] for a survey). The difference with the current work is that we do not assume that the predictions are perfect but instead have arbitrary accuracy.

When the arrival process and service times are stochastic, there are several major works that consider energy efficiency of the system. Gandhi et al. [19] provide an exact analysis of the M/M/k/setup system. The system is similar to the M/M/k class of Markov chains, i.e., tasks arrive according to a Poisson process and require an exponentially distributed processing time. To process the tasks, the system has access to a maximum of k servers. According to the algorithm in [19], if a task arrives and there are no available servers, the system moves one server to its setup state, where it remains for an exponentially random time before the server becomes active. The authors provide a sophisticated method to analyze the system exactly. Maccio and Down [37] analyze a similar system for a broader class of cost functions. When each server has a dedicated separate queue, Mukherjee et al. [44] and Mukherjee and Stolyar [45] analyze the case where the setup times and standby times (the time a server remains idle before it is deactivated) are independent exponentially distributed. In this case they propose an algorithm that achieves asymptotic optimality for both the response time and power consumption in the large-system limit. Earlier research has also modeled the response time as a constraint rather than charging a cost for the response-time [24]. Here, each task is presented with a deadline and the task should be served before this deadline or it is irrevocably lost. The earliest deadline first (EDF) queueing discipline has been proven to be effective in this case [16].

1.4 Notation and organization

The remainder of the paper is organized as follows. Section 2 describes the model. Section 3 introduces some preliminary concepts and definitions related to the ML predictions, such as the error. Section 4 introduces our algorithms and the main results, of which the high-level proof ideas are provided in Section 6. Most of the technical proofs are given in the appendix. Section 5 presents extensive numerical experiments, including the performance of our algorithms on a real-world dataset. Finally, Section 7 concludes our work and presents directions for future research.

2 Model description

We now introduce a general model for capacity scaling. Let ω, β and θ be fixed non-negative parameters of the model. We will assume that the tasks waiting in the buffer accumulate a waiting cost at rate $\omega > 0$, the cost of activating a server is $\beta > 0$, and each active server accumulates a power consumption cost at rate $\theta \geq 0$.

An instance consists of a known finite time horizon $T > 0$, a given initial number of servers $m(0) \geq 0$, and an unknown and arbitrary function $\lambda : [0, T] \rightarrow \mathbb{R}_+$ representing the arrival process. The model is:

$$\begin{aligned} & \underset{m: (0, T] \rightarrow \mathbb{R}_+}{\text{minimize}} && \omega \cdot \int_0^T q(s) \, ds + \beta \cdot \limsup_{\delta \downarrow 0} \sum_{i=0}^{\lfloor T/\delta \rfloor} [m(i\delta + \delta) - m(i\delta)]^+ + \theta \cdot \int_0^T m(s) \, ds \\ & \text{subject to} && q(t) = \int_0^t (\lambda(s) - m(s)) \mathbb{1}_{\{q(s) > 0 \text{ or } \lambda(s) \geq m(s)\}} \, ds \quad \text{for all } t \in [0, T] \\ & && m(t) \geq 0 \quad \text{for all } t \in (0, T] \end{aligned} \tag{2.1}$$

where $[x]^+ = \max(x, 0)$. To solve the optimization problem above, an algorithm needs to determine the function $m(\cdot)$, given the parameters ω, β, θ . Note that our goal is to investigate *online*

algorithms, meaning that $\lambda(\cdot)$ is revealed to the algorithm in an online fashion. In other words, at time t , the algorithm must determine $m(t)$ depending only on $\lambda(s)$ for $s \in [0, t]$. We will precisely state these assumptions later. For an algorithm that runs $m(t)$ servers at time t , the cost accumulated until time t is defined as

$$\text{Cost}^\lambda(m, t) := \omega \cdot \int_0^t q(s) \, ds + \beta \cdot \limsup_{\delta \downarrow 0} \sum_{i=0}^{\lfloor t/\delta \rfloor} [m(i\delta + \delta) - m(i\delta)]^+ + \theta \cdot \int_0^t m(s) \, ds \quad (2.2)$$

We will compare the total cost $\text{Cost}^\lambda(m, T)$ for an online algorithm to that of the *offline* minimum defined as

$$\text{OPT} := \inf_{m: (0, T] \rightarrow \mathbb{R}_+} \text{Cost}^\lambda(m, T), \quad (2.3)$$

and without loss of generality, we will assume $\text{OPT} < \infty$ throughout the paper.

Remark 2.1. The argument that minimizes (2.3) exists, as stated by the next proposition. The proof of Proposition 2.2 is given in Appendix A.1. The difficulty in the proof lies in dealing with the second term in (2.2), which makes the function $\text{Cost}^\lambda(m, T)$ discontinuous in $m(\cdot)$ w.r.t. the L_1 norm.

Proposition 2.2. *There exists $m^* : [0, T] \rightarrow \mathbb{R}_+$ such that $\text{Cost}^\lambda(m^*, T) = \text{OPT}$.*

The model in (2.1) actually combines some well-studied state-of-the-art models [1, 8, 34, 35, 37]. To see how it relates to the problem of capacity scaling, note that the objective function in (2.1) is a weighted sum of three metrics. Below, we clarify each of them. These three metrics are common performance measures of the system, such as the response time or the power consumption. The parameters ω , β and θ represent the weights assigned to each of these metrics. The three metrics are as follows:

- (i) **The flow-time.** The flow-time is defined as the total time a task spends in the system and captures the response time of the system. Note that the average response time per unit of workload is $\frac{\int_0^T q(s) \, ds}{\int_0^T \lambda(s) \, ds}$; see also [1, 8]. The weight ω is the cost attributed to the response time (e.g., in dollars per second). The weight ω could, for example, be determined based on loss of revenue or user dissatisfaction as a result of increased response time.
- (ii) **The switching cost.** As in [34, 35, 51], the parameter β can be viewed as the cost to increase the number of active servers (e.g., in dollars per server). This may include for example, the cost to terminate a lower priority service and related migration costs. In practice, these costs are usually equivalent to the cost of running the server for multiple hours [34]. The total switching cost is β times the number of times a server is made active.
- (iii) **The power consumption.** The power consumption is proportional to the total time servers are in the active state [35]. The weight θ represents the cost of power (e.g., in dollars per server per second).

Also, the constraints in (2.1) model the dynamics of capacity scaling and $q(\cdot)$ can be viewed as the queue length process or the remaining workload process. Note that (2.1) does not require $q(t)$ or $m(t)$ to be integer-valued. This is a fairly standard relaxation, since a service may typically request a fraction of the server's capacity [51, 54] and a single task is tiny; see for example, [34, 44]. The system in (2.1) can also be interpreted as a *fluid counterpart* of a discrete system. Figure 1 depicts the model schematically.

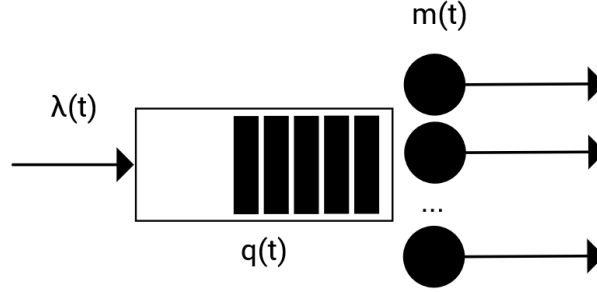


Figure 1: The system receives task at rate $\lambda(t)$ and operates $m(t)$ servers. The workload is $q(t)$.

Remark 2.3. The model in (2.1) assumes that the service capacity can be increased nearly instantaneously. Hence, it does not include the so-called setup time. Besides being a common assumption in competitive analysis (see for example [34, 35]), this is also not completely unreasonable in practice. This is mainly because servers are not usually physically turned off in reality. Instead, when a server becomes “inactive”, the server’s capacity will be used by other low-priority services. Then, activating a server means quickly terminating such low-priority services, see [51, 54] for a more detailed account. From a theoretical standpoint, the assumption of a zero setup time is also necessary to get a uniformly bounded competitive ratio, as stated in the next lemma. For the sake of Lemma 2.4, let us assume that in the capacity scaling problem in (2.1) there is an additional setup time $\tau > 0$ before the number of servers can be increased. In other words, if the online algorithm decides to turn on a server at time t , then the number of servers is increased at time $t + \tau$. The proof of Lemma 2.4 is provided in Appendix A.2.

Lemma 2.4. *Let \mathcal{A} be any deterministic algorithm for the capacity scaling problem in (2.1) and assume that there is an additional setup time $\tau > 0$ before the number of servers can be increased. Also, let CR denote the competitive ratio of \mathcal{A} . Then there exists θ such that*

$$\text{CR} \geq \frac{\omega\tau}{2\beta}, \quad (2.4)$$

In other words, there does not exist any deterministic algorithm with uniformly bounded competitive ratio.

Formulation (2.1) is fairly easy to solve as an *offline* optimization problem (in Section 5.3 we present a linear program to solve the offline problem). However, as mentioned earlier, we are interested in an *online* algorithm. Specifically, we distinguish two scenarios:

1. *Purely online scenario.* The system reveals $m(0)$, ω , β , θ and, at time t , also $\lambda(s)$ for $s \in [0, t]$ to the online algorithm, but not $\lambda(s)$ for any $s > t$. The purely online scenario corresponds to the setting where predictions may not be available and provides a natural starting point of our investigation. We discuss a competitive algorithm for the purely online scenario in Section 4.1. Additionally, in this purely online scenario, our algorithm does not require the system to reveal the finite time horizon T upfront.
2. *Machine learning scenario.* In addition to the assumptions in the purely online scenario above, at time $t = 0$, an ML predictor predicts the arrival rate function of the entire interval, that is, it predicts the arrival rate function to be $\tilde{\lambda} : [0, T] \rightarrow \mathbb{R}_+$. The ML predictor may, for example, be trained on the past observed workload on a day. For the purpose of the current work, we treat the predictor as a black box. We discuss a consistent algorithm for

the machine learning scenario in Section 4.2.1, for which the competitive ratio degrades gracefully with the prediction's accuracy. However, the algorithm is not competitive in the worst-case. Finally, in Section 4.2.2 we discuss an algorithm for the machine learning scenario, which is simultaneously competitive and consistent, by combining the algorithms from Sections 4.1 and 4.2.1.

The idea of using online algorithms with unreliable machine-learned advice was first introduced in [36] in the context of competitive caching. The next section provides the necessary details of the framework of [36].

3 Preliminary concepts

This section briefly presents the competitive analysis framework for algorithms that have access to ML predictions. The setup was first introduced in [36] and we adapt it here for the current scenario. We measure the errors in predictions by the mean absolute error (MAE) between the true and the predicted label, which is commonly used in state-of-the-art machine learning algorithms [23, 49].

Definition 3.1. *The error in the prediction $\tilde{\lambda}(\cdot)$ with respect to the actual arrival rate $\lambda(\cdot)$ is*

$$\|\tilde{\lambda} - \lambda\|_{MAE} = \frac{1}{T} \int_0^T |\tilde{\lambda}(t) - \lambda(t)| dt. \quad (3.1)$$

To measure the performance of an algorithm augmented by an ML predictor, we will define the competitive ratio as a function of the prediction's accuracy. However, before stating the definition of competitive ratio, we introduce the level of accuracy of a prediction.

Definition 3.2. *Fix a finite time horizon T , arrival rate function $\lambda(\cdot)$, and initial number of servers $m(0)$. Let OPT be as defined in (2.3). For $\eta > 0$, we say that a prediction $\tilde{\lambda}$ is η -accurate for the instance $(T, \lambda, m(0))$ if*

$$\|\tilde{\lambda} - \lambda\|_{MAE} \leq \eta \cdot \frac{\text{OPT}}{T}. \quad (3.2)$$

The definition of the prediction's accuracy is intimately tied to the cost of the optimal solution. As already argued in [36], since OPT is a linear functional of λ , normalizing the error by the cost of the optimal solution is necessary. This is because the definition should be invariant to scaling and padding arguments. For example, if we double both $\lambda(\cdot)$ and $\tilde{\lambda}(\cdot)$, then the prediction's accuracy should still be the same.

Let \mathcal{A} be any algorithm for (2.1). The performance of \mathcal{A} is measured by the competitive ratio $\text{CR}(\eta)$, which itself is a function of the accuracy η . The following definition is an adaptation of [36, Definition 3] for the current setup.

Definition 3.3. *Fix a finite time horizon T , arrival rate function $\lambda(\cdot)$, and an initial number of servers $m(0)$. Let \mathcal{A} be any algorithm for (2.1) and $m(t)$ denote its number of servers when it has access to a prediction $\tilde{\lambda}$, and OPT be as defined in (2.3). We say that \mathcal{A} has a competitive ratio at most CR for the instance $(T, \lambda, m(0))$ and prediction $\tilde{\lambda}$ if*

$$\text{Cost}^\lambda(m, T) \leq \text{CR} \cdot \text{OPT} + \Phi(0), \quad (3.3)$$

where $\Phi(0)$ is a constant depending only on $m(0)$. We say that the competitive ratio of \mathcal{A} is at most $\text{CR}(\eta)$, if the competitive ratio is at most $\text{CR}(\eta)$ for all instances $(T, \lambda, m(0))$ and any η -accurate prediction $\tilde{\lambda}$.

Note that, although the competitive ratio depends on the prediction's accuracy, the algorithm is oblivious to this accuracy. We desire three properties of an online algorithm that has access to a prediction. The algorithm's performance should (i) be close to the optimal solution if the prediction is perfect, (ii) degrade gracefully with the prediction's error, and (iii) be bounded regardless of the prediction's accuracy. The definitions of consistency, robustness, and competitiveness summarize these desiderata; see also [36, Definition 4, 5, and 6].

Definition 3.4. Let \mathcal{A} be any algorithm for (2.1), and $\text{CR}(\eta)$ denote its competitive ratio when it has access to an η -accurate prediction. Then, we say that algorithm \mathcal{A} is

- (i) ρ -consistent if $\text{CR}(0) = \rho$;
- (ii) α -robust if $\text{CR}(\eta) = O(\alpha(\eta))$;
- (iii) γ -competitive if $\text{CR}(\eta) \leq \gamma$ for all $\eta \in [0, \infty]$.

4 Main results

4.1 Balanced Capacity Scaling algorithm

We first discuss a competitive algorithm in the purely online scenario. Recall that in this case the system reveals $m(0)$, ω , β , θ and, at time t , also $\lambda(s)$ for $s \in [0, t]$ to the algorithm, but not $\lambda(s)$ for any $s > t$. Moreover, as mentioned earlier, the results in this section also hold when the finite time horizon T is not revealed upfront. The Balanced Capacity Scaling (BCS) algorithm that we propose is parameterized by two non-negative numbers r_1 and r_2 . Algorithm 1 below describes BCS for any fixed choices of r_1 and r_2 .

Algorithm 1: BCS (r_1, r_2)

Choose $m(\cdot)$ such that at each time $t \geq 0$:

$$\frac{dm(t)}{dt} = \frac{r_1 \omega \cdot q(t) - r_2 \theta \cdot m(t)}{\beta}. \quad (4.1)$$

We start by briefly discussing the intuition behind BCS. At each time $t \geq 0$, BCS computes the derivative of the number of servers, i.e., how fast the system should increase or decrease the service capacity. Note that if we solve equation (4.1), then we obtain the number of servers $m(t)$, which is differentiable for all $t \geq 0$. The two parameters r_1 and r_2 control how fast the algorithm reacts, by increasing or decreasing the number of servers respectively. If the workload $q(t)$ is non-zero, then the first term in the right-hand side of equation (4.1) increases the number of servers at rate r_1 . The second term is an “inertia term” which decreases the number of servers at rate r_2 . Note that if we integrate equation (4.1), we obtain

$$\int_0^t r_1 \omega \cdot q(s) ds = \int_0^t \beta \cdot \frac{dm(s)}{ds} ds + \int_0^t r_2 \theta \cdot m(s) ds. \quad (4.2)$$

This means that BCS aims to carefully balance the flow-time with the switching cost plus the power consumption. The BCS algorithm is memoryless and computationally cheap. The derivative of the number of servers depends only on the current workload and number of servers, without requiring knowledge about the past workload, number of servers or arrival rate. BCS can therefore

be implemented without any memory requirements.

We are able to characterize the performance of BCS analytically, for any fixed choices of r_1 and r_2 . Theorem 4.1 below characterizes the competitive ratio of the BCS algorithm. The proof of Theorem 4.1 is provided in Section 6.1.

Theorem 4.1. *Let CR denote the competitive ratio of BCS (Algorithm 1). Then,*

$$\text{CR} \leq \left(1 + \frac{1}{r_1} + \frac{1}{r_2}\right) \max(2, r_1, 2r_2). \quad (4.3)$$

The optimal choice of the parameters is $r_1 = 2$ and $r_2 = 1$. Corollary 4.2 states that BCS is 5-competitive in this case.

Corollary 4.2. *Let CR denote the competitive ratio of BCS (Algorithm 1). If $r_1 = 2$ and $r_2 = 1$, then $\text{CR} \leq 5$.*

Moreover, in the special case when tasks are not allowed to wait and must be served immediately upon arrival ($\omega = \infty$), BCS turns out to be 2-competitive, as stated in Theorem 4.3. The proof of Theorem 4.3 is given in Appendix A.3.

Theorem 4.3. *Let CR denote the competitive ratio of BCS (Algorithm 1). If $r_1 = 2$, $r_2 = 1$ and $\omega = \infty$, then $\text{CR} \leq 2$.*

Note that the capacity scaling problem has previously been related to the classical ski-rental problem [5, 27, 35] which is 2-competitive. As it turns out, when tasks are allowed to wait, the formulation in (2.1) of the capacity scaling problem is strictly harder than the ski-rental problem, as Proposition 4.4 states below. Proposition 4.4 is proved in Appendix A.4.

Proposition 4.4. *Let \mathcal{A} be any deterministic algorithm for the capacity scaling problem in (2.1) in the purely online scenario, and CR denote its competitive ratio. There exist choices for ω , β , and θ such that $\text{CR} \geq 2.549$. In other words, any deterministic algorithm is at least 2.549-competitive.*

Remark 4.5. We should note that the proof of Proposition 4.4 assumes that the finite time horizon T is not revealed upfront. We leave it to future work to identify a (possibly weaker) lower bound if T is known to the algorithm.

4.2 Augmenting unreliable ML predictions

To augment the BCS algorithm with machine learning predictions, we proceed in two steps. First, in section 4.2.1, we introduce the Adapt to the Prediction (AP) algorithm. We prove that the competitive ratio of AP degrades gracefully with the prediction's accuracy, although AP is not competitive. Second, in section 4.2.2, we discuss how to combine the BCS and AP algorithms to obtain the ABCS algorithm, which follows the predictions but is robust against inaccurate predictions and therefore competitive.

4.2.1 Adapt to the Prediction algorithm

We will now turn our attention to the machine learning scenario. Recall that in this case, at time $t = 0$, the algorithm receives a predicted arrival rate function $\tilde{\lambda} : [0, T] \rightarrow \mathbb{R}_+$. Note that a trivial way to implement the predictions is to blindly trust the predictions, i.e., to let

$$m \in \arg \min_{m: (0, T] \rightarrow \mathbb{R}_+} \text{Cost}^{\tilde{\lambda}}(m, T). \quad (4.4)$$

The above minimum exists (see Remark 2.1). However, in this case, the performance decays drastically even for relatively small prediction errors. Indeed, if the actual arrival rate $\lambda(\cdot)$ is higher than the predicted arrival rate $\tilde{\lambda}(\cdot)$ at the start, then the associated workload could stay in the queue until the end of the time horizon $[0, T]$ and incur a significant waiting cost. We instead propose the Adapt to the Prediction (AP) algorithm, which consists of an offline and an online component. The offline component computes an estimate for the number of servers upfront based on $\tilde{\lambda}(\cdot)$. The online component follows the offline estimate, but dynamically adapts the number of servers based on discrepancies between the predicted and actual arrival rates. Let us define

$$\Delta\lambda(t) := \begin{cases} (\lambda(t) - \tilde{\lambda}(t))^+ & \text{for } t \geq 0 \\ 0 & \text{for } t < 0 \end{cases}$$

Algorithm 2 below describes the AP algorithm.

Algorithm 2: AP

Choose $m(\cdot)$ such that at each time $t \geq 0$:

$$m(t) = m_1(t) + m_2(t), \tag{4.5}$$

where

$$m_1 \in \arg \min_{m: (0, T] \rightarrow \mathbb{R}_+} \text{Cost}^{\tilde{\lambda}}(m, T), \tag{4.6}$$

$$\frac{dm_2(t)}{dt} = \sqrt{\frac{\omega}{2\beta}} \cdot \left(\Delta\lambda(t) - \Delta\lambda\left(t - \sqrt{2\beta/\omega}\right) \right). \tag{4.7}$$

The number of servers under the AP algorithm consists of two components, an offline component m_1 and an online component m_2 . The offline component m_1 is determined upfront by the optimal number of servers to handle the predicted arrival rate $\tilde{\lambda}$. The online component m_2 is determined in an online manner and it reacts if the actual arrival rate turns out to be higher than the predicted arrival rate. Note that if we solve equation (4.7), then we obtain the number of servers $m_2(t)$, which is differentiable for all $t \geq 0$. The online component works as follows. If $\Delta\lambda(t) > 0$, then the online component increases the service capacity at rate $\sqrt{\omega/(2\beta)}$. In other words, for each additional unit of workload received, the number of servers is increased by $\sqrt{\omega/(2\beta)}$. After a fixed time of $\sqrt{2\beta/\omega}$ the number of servers is decreased again. Intuitively, if $\omega \gg \beta$, then the online component turns on many servers for a short period of time, whereas if $\beta \gg \omega$, then the online component turns on a few servers for a longer period of time. The constants are chosen to carefully balance the flow-time with the switching cost.

Although the optimization in the offline component might be expensive, it has to be performed only once at the start. Moreover, if the predictions are based on historical data, the offline component m_1 might even be precomputed and retrieved from memory at the start. The online component in contrast is computationally cheap.

The competitive ratio of AP, of course, depends on the accuracy of the predictions. Theorem 4.6 characterizes the performance of AP. Recall the definition of the competitive ratio $\text{CR}(\eta)$ from Section 3.

Theorem 4.6. Let $\text{CR}(\eta)$ denote the competitive ratio of AP (Algorithm 2) when it has access to an η -accurate prediction. Then,

$$\text{CR}(\eta) \leq 1 + \left(\sqrt{2\omega\beta} + \theta \right) \eta. \quad (4.8)$$

The proof of Theorem 4.6 is provided in Appendix A.5. If η is small, then the competitive ratio is close to one. In fact, the AP algorithm replicates the optimal solution exactly (in L_1 sense) if the predictions turn out to be accurate and is 1-consistent. Moreover, the competitive ratio also degrades gracefully in the prediction's accuracy, which, as discussed earlier, is not achieved by the offline component m_1 alone.

Remark 4.7. Although AP does not follow the predictions blindly, the AP algorithm is not competitive, since it is not hard to verify that $\text{CR}(\eta) \rightarrow \infty$ as $\eta \rightarrow \infty$ (e.g. let $\tilde{\lambda}(t) \rightarrow \infty$ for all $t \in [0, T]$). Note that most algorithms proposed in the literature, such as the RHC and LCP algorithms from [34], follow the predictions blindly and therefore are not competitive. Hence, the goal in the next subsection is to combine the above approaches of BCS and AP to obtain an algorithm which follows the predictions most of the time, but ignores the predictions when appropriate.

4.2.2 Adaptive Balanced Capacity Scaling

We now answer the question of how to follow the predictions most of the time without trusting them blindly. The Adaptive Balanced Capacity Scaling (ABCS) algorithm we propose, strategically combines the BCS and AP algorithms introduced earlier. Let $\tilde{m}(\cdot)$ be the number of servers as turned on by AP (Algorithm 2). Let $\tilde{q}(\cdot)$ be the queue length process under the AP algorithm, that is,

$$\tilde{q}(t) = \int_0^t (\lambda(s) - \tilde{m}(s)) \mathbb{1}_{\{\tilde{q}(s) > 0 \text{ or } \lambda(s) \geq \tilde{m}(s)\}} ds \quad \text{for all } t \geq 0. \quad (4.9)$$

The ABCS algorithm is parameterized by four non-negative numbers $R_1 \geq r_1 \geq 0$ and $R_2 \geq r_2 \geq 0$. Algorithm 3 below describes ABCS for any fixed choices of R_1, r_1, R_2, r_2 .

Algorithm 3: ABCS (r_1, r_2, R_1, R_2)

Choose $m(\cdot)$ such that at each time $t \geq 0$:

$$\frac{dm(t)}{dt} = \frac{\hat{r}_1(t)\omega \cdot q(t) - \hat{r}_2(t)\theta \cdot m(t)}{\beta}, \quad (4.10)$$

where

$$\begin{aligned} \hat{r}_1(t) &= \begin{cases} r_1 & \text{if } m(t) - \tilde{m}(t) > [q(t) - \tilde{q}(t)]^+ \cdot \sqrt{\frac{\omega}{2\beta}}, \\ R_1 & \text{if } m(t) - \tilde{m}(t) \leq [q(t) - \tilde{q}(t)]^+ \cdot \sqrt{\frac{\omega}{2\beta}}, \end{cases} \\ \hat{r}_2(t) &= \begin{cases} R_2 & \text{if } m(t) > \tilde{m}(t) \text{ and } q(t) \leq \tilde{q}(t), \\ r_2 & \text{if } m(t) \leq \tilde{m}(t) \text{ or } q(t) > \tilde{q}(t). \end{cases} \end{aligned} \quad (4.11)$$

In spirit, the ABCS algorithm works similarly to the BCS algorithm. In fact, if $R_1 = r_1$ and $R_2 = r_2$, then the ABCS algorithm is equivalent to the BCS algorithm and disregards predictions altogether. However, in contrast to the constant rates r_1 and r_2 of BCS, the rates at which ABCS reacts, is captured by the state-dependent rate functions $\hat{r}_1(t)$ and $\hat{r}_2(t)$. The reason behind the

precise choices of $\hat{r}_1(t)$ and $\hat{r}_2(t)$ will be clear later from the performance of the algorithm. From a high-level perspective, these are chosen to approach the behavior of the advised number of servers $\tilde{m}(t)$ of AP. Indeed, if ABCS has less than the advised number of servers $\tilde{m}(t)$, then it increases $m(t)$ at the higher rate R_1 and decreases it at the lower rate r_2 . Similarly, if ABCS has “sufficiently more” servers than the advised number $\tilde{m}(t)$, then it increases $m(t)$ at the lower rate r_1 and decreases it at the higher rate R_2 . The number of servers of ABCS therefore judiciously approaches the number of advised servers. However, it does not blindly follow $\tilde{m}(t)$ to protect against inaccurate predictions. For example, if the workload $q(t)$ is significantly higher than the current number of servers $m(t)$, then ABCS will always increase the number of servers at a non-zero rate.

Our main result characterizes the performance of ABCS analytically, which is presented in Theorem 4.8 below. The proof of Theorem 4.8 is provided in Section 6.2. Recall the definition of the competitive ratio $\text{CR}(\eta)$ from Section 3.

Theorem 4.8. *Let $\text{CR}(\eta)$ denote the competitive ratio of ABCS (Algorithm 3) when it has access to an η -accurate prediction. Then,*

$$\text{CR}(\eta) \leq \min \left((1 + (\sqrt{2\omega\beta} + \theta)\eta) \cdot \text{OCR}, \text{PCR} \right), \quad (4.12)$$

where

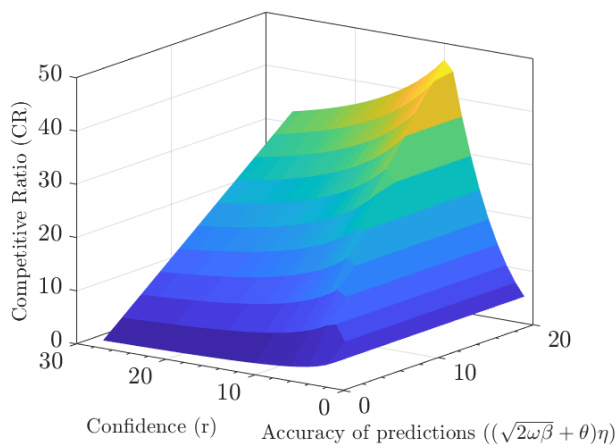
$$\begin{aligned} \text{OCR} &= \max \left(c_1 r_1, \frac{c_2 R_1}{\sqrt{1 + 2R_1}}, c_2 + c_3, c_4 \right), \quad \text{PCR} = \max \left(c_5 R_1, 2c_6, 2c_6 R_2 + 1 - \frac{R_2}{r_2} \right), \\ c_1 &= 1 + \frac{1}{r_1} + \frac{1}{R_2}, \quad c_2 = \frac{c_1 \sqrt{1 + 2r_1} - c_1 + c_3}{\sqrt{1 + 2R_1}}, \quad c_3 = 1 + \frac{1}{R_1} + \frac{1}{R_2}, \\ c_4 &= 1 + r_2 + \frac{r_2}{R_1} + c_2 r_2, \quad c_5 = 1 + \frac{1}{r_1} + \frac{1}{r_2}, \quad c_6 = c_5 \sqrt{\frac{R_1}{r_1}}. \end{aligned} \quad (4.13)$$

Theorem 4.8 characterizes the competitive ratio of ABCS explicitly for any choices of the parameters. Note that for any value of η , the competitive ratio is at most PCR. Moreover, if η is small, then the competitive ratio is close to OCR. It is straightforward to check from Theorem 4.8 that ABCS satisfies the three desiderata of Definition 3.4. In particular, ABCS is OCR-consistent and PCR-competitive. The constants OCR and PCR, of course, depend on the parameters R_1 , r_1 , R_2 and r_2 . Corollary 4.9 provides guidance on how to choose these parameters. For ease of understanding, we have stated Corollary 4.9 using asymptotic comparison symbols.

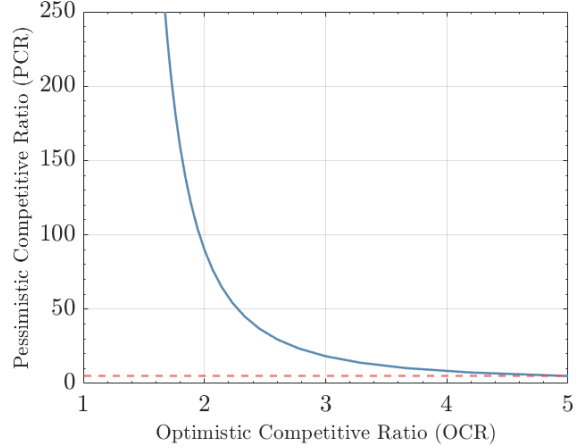
Corollary 4.9. *Let $r \geq 1$ be a hyperparameter, representing the confidence in the predictions. Let $\text{CR}(\eta)$ denote the competitive ratio of ABCS (Algorithm 3) when it has access to an η -accurate prediction. If $R_1 = \Theta(r)$, $r_1 = \Theta(r^{-1})$, $R_2 = \Theta(r)$ and $r_2 = \Theta(r^{-1})$, then*

$$\text{CR}(\eta) \leq \min \left((1 + (\sqrt{2\omega\beta} + \theta)\eta) \cdot (1 + \Theta(r^{-1})), \Theta(r^3) \right). \quad (4.14)$$

Corollary 4.9 characterizes the trade-off between the OCR and the PCR. If the confidence in the predictions, r , is set at a high value, then OCR tends to 1. However, the value of PCR tends to become large in this case, even though, importantly, it remains uniformly bounded as $\eta \rightarrow \infty$. Figure 2a plots the competitive ratio as a function of η and the confidence hyperparameter r . For fixed r , the competitive ratio increases linearly in η . If η is large enough, then the competitive ratio becomes constant in η at a value of PCR. On the other hand, for fixed η , the competitive ratio is always 5 in the case of zero confidence ($R_1 = r_1$ and $R_2 = r_2$). Then, as the confidence increases, the competitive ratio decreases slowly if η is small enough and increases rapidly if η is large.



(a) The competitive ratio as a function of the normalized accuracy of the predictions $(\sqrt{2\omega\beta} + \theta)\eta$ and the confidence hyperparameter r . The competitive ratio decreases as predictions are more accurate, but remains bounded as the accuracy becomes worse.



(b) The Pessimistic Competitive Ratio (PCR) as a function of the Optimistic Competitive Ratio (OCR). The figure interpolates between the purely online scenario (OCR = PCR = 5) and the machine learning scenario (OCR = 1 and PCR = ∞).

Figure 2: The analytical performance of ABCS (Algorithm 3).

Remark 4.10. Figure 2b plots the value of PCR on the y-axis, for a fixed OCR on the x-axis. Figure 2b depicts the interpolation between the purely online scenario (OCR = PCR = 5) and the machine learning scenario (OCR = 1 and PCR = ∞). The current work generalizes these two extremes to any scenario in between. As mentioned in the introduction, we provide performance guarantees for ABCS for any value of the confidence hyperparameter $r \geq 1$. However, the specific choice of r would depend on where the system designer wants to place the system on the blue curve in Figure 2b; view it as a risk-vs-gain curve. The figure shows that if one chooses a value of r , so that if the predictions turn out to be accurate, ABCS would be 2-competitive, then that would put the system at the risk of being up to about 90-competitive, if the predictions turn out to be completely wrong. Later, in Proposition 4.12 we show that the growth rate of the OCR-vs-PCR curve that we obtain for ABCS is nearly optimal in the sense that any algorithm which is $(1 + \delta)$ -competitive in the optimistic case must be at least $1/(4\delta^2)$ -competitive in the pessimistic case.

Remark 4.11. Recently, there has been some interest in understanding the performance of algorithms when an estimate of the prediction's accuracy η is available in terms of some probability distribution [42, 43]. In such cases, Theorem 4.8 allows one to calculate the optimal choice of confidence hyperparameter r that minimizes the expected competitive ratio. Assume that the prediction's accuracy η follows some distribution $\mu(\cdot)$. The distribution $\mu(\cdot)$ might, for example, be estimated based on historically observed data. For a fixed r , note that OCR and PCR are functions of r . Denote

$$\zeta(r) := \frac{\text{PCR} - \text{OCR}}{2\text{OCR}}.$$

The expected value of the *random* competitive ratio of ABCS is then

$$\begin{aligned}
\mathbb{E}_{\eta \sim \mu}[\text{CR}(\eta)] &= \int_0^\infty \min((1 + 2\eta) \cdot \text{OCR}, \text{PCR}) d\mu(\eta) \\
&= 2\text{OCR} \cdot \int_0^{\zeta(r)} \eta d\mu(\eta) + \text{OCR} \cdot \int_0^{\zeta(r)} d\mu(\eta) + \text{PCR} \cdot \int_{\zeta(r)}^\infty d\mu(\eta) \\
&= 2\text{OCR} \cdot \mathbb{E}[\eta \mathbb{1}\{\eta \leq \zeta(r)\}] + \text{OCR} \cdot \mathbb{P}(\eta \leq \zeta(r)) + \text{PCR} \cdot \mathbb{P}(\eta > \zeta(r)).
\end{aligned} \tag{4.15}$$

Therefore, if either the distribution or an estimate thereof is known, then the parameters of ABCS can be chosen to minimize the expected competitive ratio.

Theorem 4.8 and Corollary 4.9 demonstrate that there is a trade-off between the OCR and the PCR. The following proposition shows that this trade-off is in fact, inherent to the problem and is not an artifact of the algorithm.

Proposition 4.12. *Let \mathcal{A} be any deterministic algorithm for the capacity scaling problem in (2.1), and $\text{CR}(\eta)$ denote its competitive ratio when it has access to an η -accurate prediction. There exist choices of ω , β , and θ such that, for any $\delta > 0$, if $\text{CR}(0) \leq 1 + \delta$, then*

$$\text{CR}\left(\frac{2}{\delta^2}\right) \geq \frac{1}{4\delta^2}. \tag{4.16}$$

In other words, any deterministic algorithm that is $(1 + \delta)$ -consistent must be $\Omega(1/\delta^2)$ -competitive.

Proposition 4.12 is proved in Appendix A.9. As mentioned earlier, an algorithm for capacity scaling must consist of two components: one component is to decide when to activate a server and the other component is to decide when to deactivate a server. For the latter problem, a popular state-of-the-art approach is to implement a power-down timer [5, 19, 27, 29, 35, 44]. The power-down timer works as follows: each time a server becomes idle, the system starts a timer corresponding to that server. If the server remains idle after the timer expires, then the server is deactivated. Algorithm 4 shows the Timer algorithm.

Algorithm 4: The Timer Algorithm

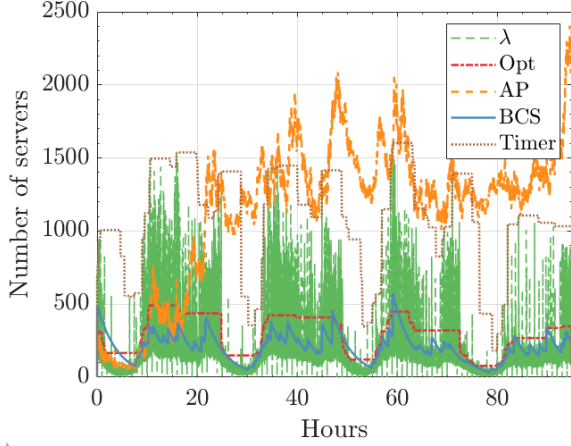
At each time $t \geq 0$:

Turn off a server if the server has been idle for more than β/θ time.

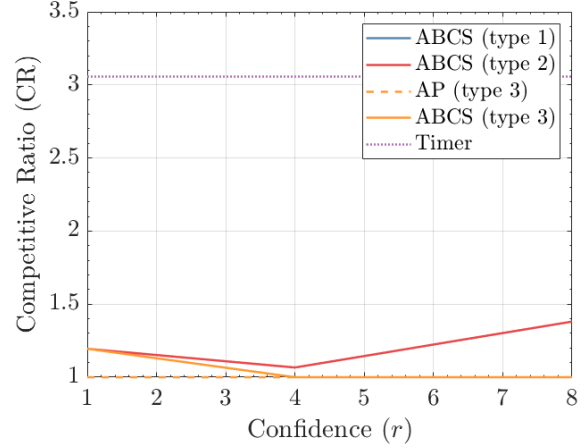
We end this section by pointing out that although the Timer algorithm above has proven to be successful under specific (especially stochastic) scenarios, the worst-case performance of the algorithm in the current context is poor as the following proposition shows. More specifically, Proposition 4.13 shows that the Timer algorithm does not have uniformly bounded competitive ratio. To the best of our knowledge, there does not exist any uniformly competitive algorithm for the capacity scaling problem where ω is finite, until in the current work. Proposition 4.13 is proved in Appendix A.10.

Proposition 4.13. *Fix any rule for activating servers and any $\omega > 0$. There exist choices of β and θ such that the competitive ratio of the Timer algorithm (Algorithm 4) is at least $(2\omega)^{-1}$.*

Remark 4.14. The constant β/θ is the default choice if $\omega = \infty$ and is used in popular state-of-the-art approaches [5, 27, 29, 34]. The result of Proposition 4.13 extends readily to any constant multiple of β/θ . However, for other choices, the competitive ratio depends on the rule for activating servers.



(a) The real-world arrival pattern (λ) across four consecutive days and the number of servers of Opt, AP, BCS and the Timer algorithm.



(b) The competitive ratio (CR) of AP, ABCS and the Timer algorithm as a function of the confidence hyperparameter (r). The CR of AP and ABCS further depends on the type of predictions provided (type 1-4). The CR of AP for type 1 and 2 is at least 20 and has not been presented here. The CR of ABCS for type 1 corresponds with the CR of ABCS for type 3.

Figure 3: The performance of our algorithms evaluated on a real-world dataset.

5 Numerical experiments

5.1 Real-world dataset

We first verify the performance of the proposed algorithms on a real-world dataset of internet traffic. The dataset consists of DNS requests observed at a campus network across four consecutive days in April 2016 [39]. We extracted the number of requests at each second to obtain the arrival rate function $\lambda(t)$. Furthermore, we assume that each server consumes 850 W at a price of 0.15 cents per kWh, β is equal to the power-cost of running a server for four hours and $\omega = 0.1$ cents. Figure 3a presents the arrival rate function and an example run of AP (Algorithm 2), BCS (Algorithm 1) and the Timer algorithm (Algorithm 4). We will investigate the influence of the weights ω , β and θ and the confidence hyperparameter r on the performance. Moreover, we will test the influence of the predictions on the performance of ABCS and AP. The results in this section will illustrate the performance of these algorithms in practice as opposed to the worst-case results presented before.

The real-world arrival pattern fluctuates between 0-20 requests per second during night time and up to 1500 requests per second at peak hours. Figure 4 presents the competitive ratios of AP (Algorithm 2), ABCS (Algorithm 3), and the Timer algorithm (Algorithm 4) as a function of the weights ω , β and θ . AP and ABCS further depend on the predictions provided. Let $\lambda(\cdot)$ be the arrival rate. The type of predictions used are as follows:

- *Type 1.* The system does not reveal any predictions, i.e. $\tilde{\lambda}(t) = 0$ for all $t \in [0, T]$.
- *Type 2.* The system reports the moving average (MA) across three hours, i.e. $\tilde{\lambda}(t) = (\min(t, 1.5) + \min(T - t, 1.5))^{-1} \int_{\max(t-1.5, 0)}^{\min(t+1.5, T)} \lambda(t) dt$ for all $t \in [0, T]$.
- *Type 3.* The system provides perfect predictions, i.e. $\tilde{\lambda}(t) = \lambda(t)$ for all $t \in [0, T]$.

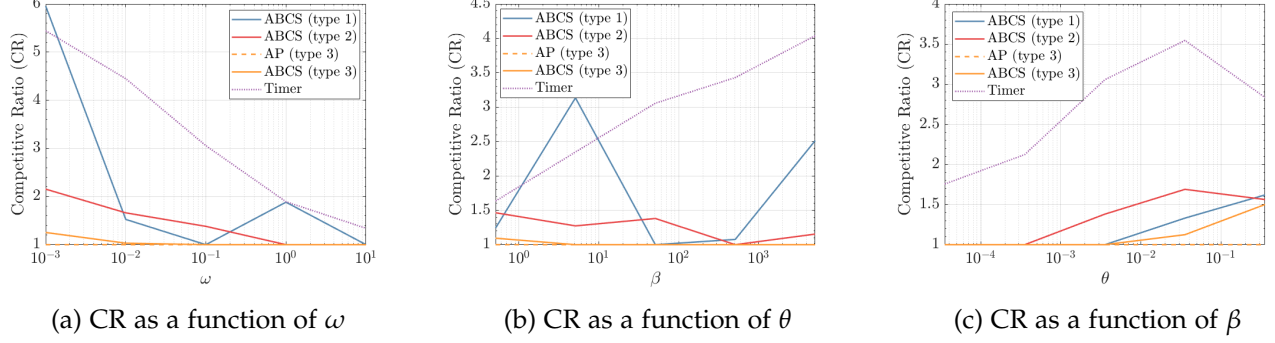


Figure 4: The competitive ratio (CR) of AP, ABCS, and the Timer algorithm as a function of the weights ω , β and θ for a real-world arrival pattern. The confidence hyperparameter of ABCS is $r = 8$. The CR of AP and ABCS further depends on the type of predictions provided (type 1-3). The CR of AP for type 1 and 2 is at least 20 and has not been presented here.

The AP algorithm performed poorly for type 1 and type 2 and its competitive ratio has not been presented in Figure 4. The poor performance was already expected from Theorem 4.6 because the prediction’s accuracy is low. ABCS has a significantly lower competitive ratio than AP for inaccurate predictions (type 1 and type 2), proving that ABCS is robust against inaccurate predictions. Moreover, even if only the moving average (MA) is available, the performance of ABCS improves in most cases. The competitive ratio of ABCS if accurate predictions are available (type 3) is close to 1 for any choice of weights. Also, note that ABCS outperforms the Timer algorithm for almost any type of prediction and choice of weights.

Figure 3b presents the competitive ratio of the AP (Algorithm 2), ABCS (Algorithm 3) and the Timer algorithm (Algorithm 4) as a function of the confidence hyperparameter (r) as introduced in Corollary 4.9. The competitive ratio only slightly depends on the confidence hyperparameter. The competitive ratio of ABCS for type 1 matches the competitive ratio for type 3, which is surprising and generally not true in light of Figure 4. The competitive ratio of ABCS increases as the confidence increases if inaccurate predictions are available (type 2) and decreases if accurate predictions are available (type 3).

5.2 Artificial data

To empirically verify the performance against extreme cases, we tested the proposed algorithms on three artificially generated datasets. The arrival rate functions from these datasets are not intended to model the real-world internet traffic. Rather, the arrival rate functions are highly stylized versions of specific patterns occurring in real-world traffic. The three patterns considered are a sinusoidal, a sharp peak, and a step-function as seen in Figure 5. Figure 5 also presents an example run of the AP (Algorithm 2), BCS (Algorithm 1), and the Timer algorithm (Algorithm 4). The results will provide insights into how our algorithm respond to features of the demand process. For example, the sinusoidal represents an average of the arrival rate during a day, while the sharp peak may represent an unpredictable surge in workload. For each pattern, we will investigate the influence of the weights ω , β and θ and the confidence hyperparameter r on the performance. Moreover, for each pattern, we have further identified three to four predicted patterns to investigate the influence of the predictions on the performance.

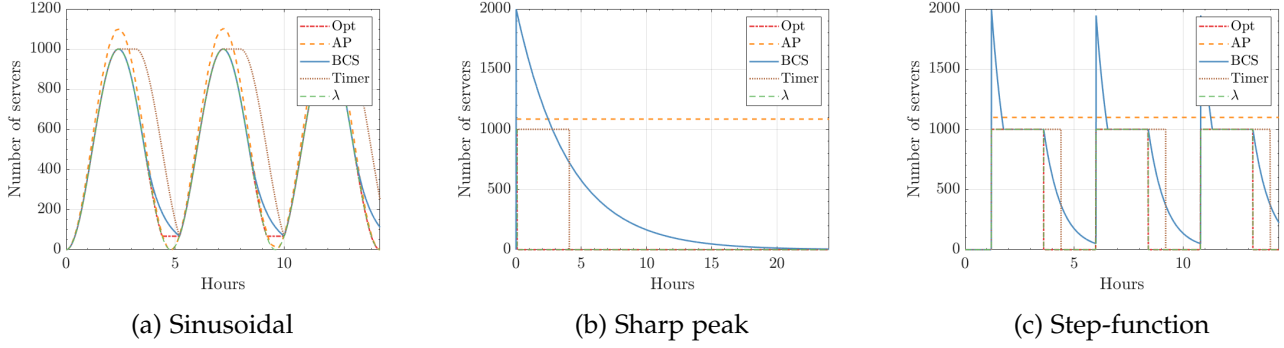


Figure 5: The three artificial arrival patterns (λ) considered and the number of servers of Opt, AP, BCS and the Timer algorithm.

Sinusoidal. The sinusoidal pattern represents a stylized periodic arrival rate function. The sinusoidal pattern has a period of 24 hours, varying between $\lambda(t) = 0$ and $\lambda(t) = 1000$. Figure 6 presents the competitive ratio of the AP (Algorithm 2), ABCS (Algorithm 3) and the Timer algorithm (Algorithm 4) as a function of the weights ω , β and θ . AP and ABCS further depend on the predictions provided. Let $\lambda(\cdot)$ be the arrival rate. The type of predictions are as follows:

- *Type 1.* The system does not reveal any predictions, i.e. $\tilde{\lambda}(t) = 0$ for all $t \in [0, T]$.
- *Type 2.* The system predicts only the average of the arrival rate, i.e. $\tilde{\lambda}(t) = 500$ for all $t \in [0, T]$.
- *Type 3.* The system predicts the opposite of the arrival rate, i.e. $\tilde{\lambda}(t) = 1000 - \lambda(t)$ for all $t \in [0, T]$.
- *Type 4.* The system provides perfect predictions, i.e. $\tilde{\lambda}(t) = \lambda(t)$ for all $t \in [0, T]$.

The competitive ratio of AP decreases as more accurate predictions are available. Generally, the availability of perfect predictions (type 4) guarantees a competitive ratio of 1, while the competitive ratio of opposite predictions (type 3) is higher than not revealing any predictions at all (type 1). Whether the competitive ratio increases or decreases if the average of the arrival rate (type 2) is available depends on the weights. The ABCS algorithm is more robust against inaccurate predictions; the competitive ratio never exceeds 1.45 for any type of prediction or choice of weights. For this pattern, ABCS is able to closely replicate the optimal solution, even without the availability of predictions. Also, ABCS performs better than the Timer algorithm for almost any choice of weights.

Sharp peak. The sharp peak pattern represents a (potentially unpredictable) surge in workload that lasts for a short time. The sharp peak lasts five minutes at $\lambda(t) = 1000$. Figure 7 presents the competitive ratio of the AP (Algorithm 2), ABCS (Algorithm 3) and the Timer algorithm (Algorithm 4) as a function of the weights ω , β and θ . The type of predictions are as follows:

- *Type 1.* The system does not reveal any predictions, i.e. $\tilde{\lambda}(t) = 0$ for all $t \in [0, T]$.
- *Type 2.* The system predicts only 50% of the peak height, i.e. $\tilde{\lambda}(t) = 0.5 \cdot \lambda(t)$ for all $t \in [0, T]$.
- *Type 3.* The system provides perfect predictions, i.e. $\tilde{\lambda}(t) = \lambda(t)$ for all $t \in [0, T]$.

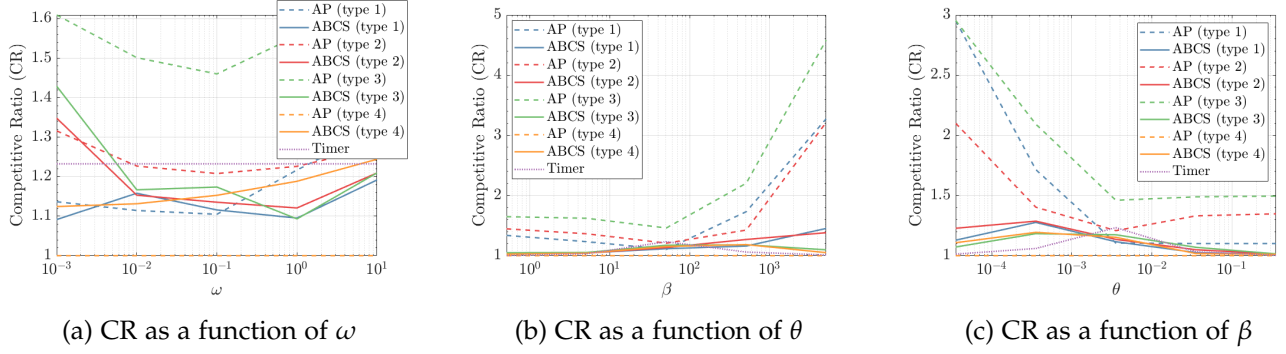


Figure 6: The competitive ratio (CR) of AP, ABCS and the Timer algorithm as a function of the weights ω , β and θ for a sinusoidal arrival pattern. The confidence hyperparameter of ABCS is $r = 8$. The CR of AP and ABCS further depends on the type of predictions provided (type 1-4).

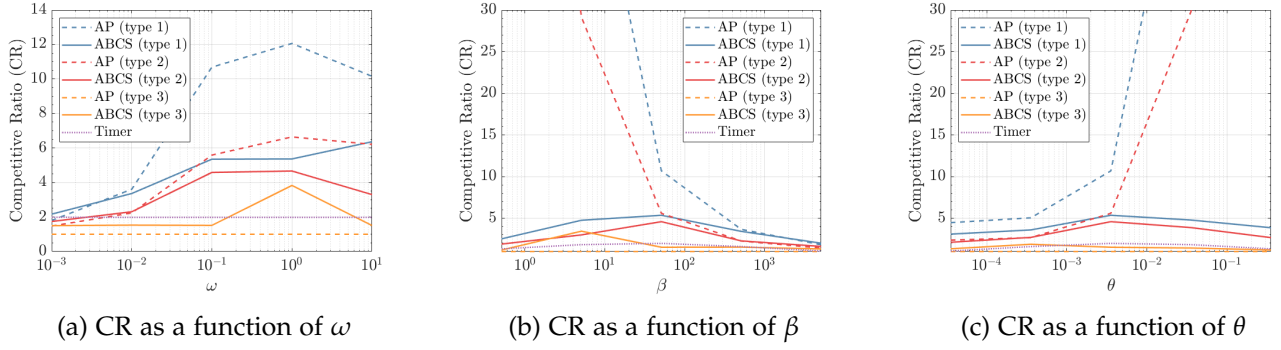


Figure 7: The competitive ratio (CR) of AP, ABCS and the Timer algorithm as a function of the weights ω , β and θ for a sharp peak arrival pattern. The confidence hyperparameter of ABCS is $r = 8$. The CR of AP and ABCS further depends on the type of predictions provided (type 1-3).

The competitive ratio of both AP and ABCS decreases as more accurate predictions are available. The competitive ratio is highest if the system does not reveal any predictions (type 1), followed by if the system predicts only 50% of the peak height (type 2) and if predictions are perfect (type 3). Moreover, if the predictions are inaccurate (type 1 and type 2), then ABCS significantly outperforms AP and hence is robust against unpredictable surges in workload. Still, if the predictions are accurate (type 3), then ABCS performs close to the optimal solution for almost any choice of weights.

Step-function. The step-function pattern is another periodic arrival rate function with sudden zero-one steps. The step-function pattern has a period of 24 hours, alternating between $\lambda(t) = 1000$ for 12 hours and $\lambda(t) = 0$ for 12 hours. Figure 8 presents the competitive ratio of the AP (Algorithm 2), ABCS (Algorithm 3) and the Timer algorithm (Algorithm 4) as a function of the weights ω , β and θ . The type of predictions are similar to the sinusoidal pattern as follows:

- *Type 1.* The system does not reveal any predictions, i.e. $\tilde{\lambda}(t) = 0$ for all $t \in [0, T]$.
- *Type 2.* The system predicts only the average of the arrival rate, i.e. $\tilde{\lambda}(t) = 500$ for all $t \in [0, T]$.

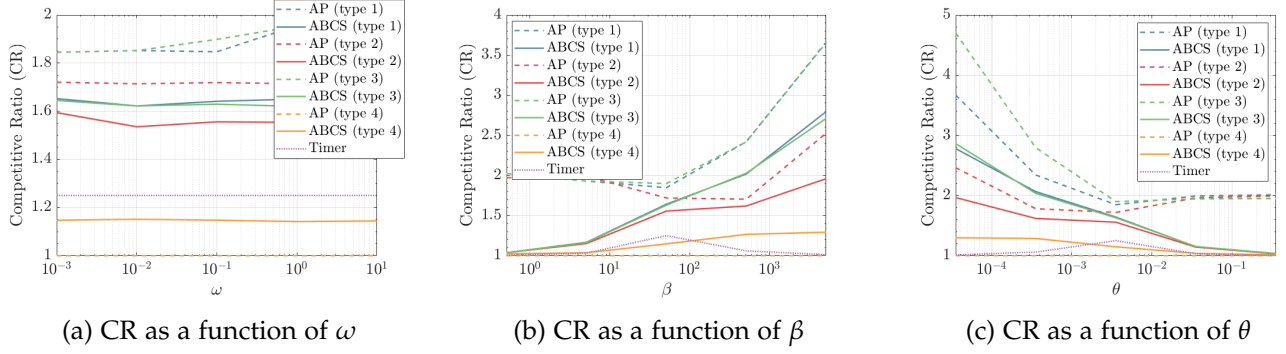


Figure 8: The competitive ratio (CR) of AP, ABCS and the Timer algorithm as a function of the weights ω , β and θ for a step-function arrival pattern. The confidence hyperparameter of ABCS is $r = 8$. The CR of AP and ABCS further depends on the type of predictions provided (type 1-4).

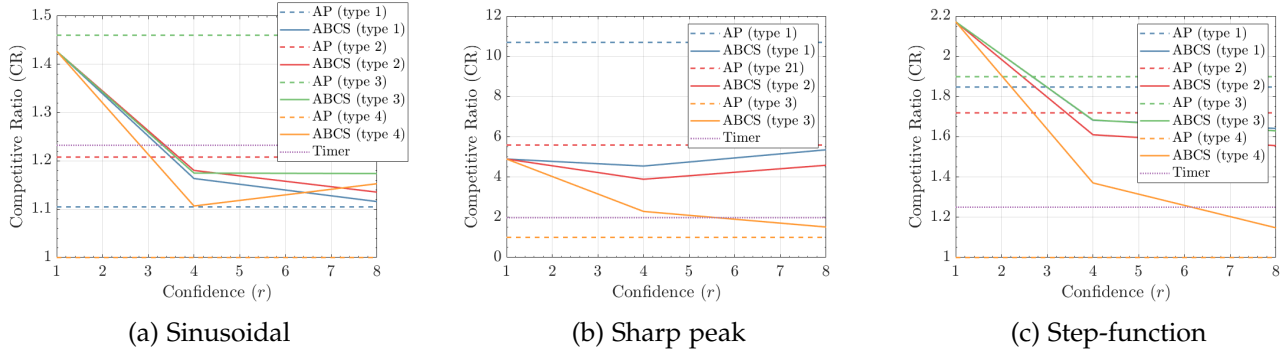


Figure 9: The competitive ratio (CR) of AP, ABCS and the Timer algorithm as a function of the confidence hyperparameter (r). The CR of AP and ABCS further depends on the type of predictions provided (type 1-4).

- *Type 3.* The system predicts the opposite of the arrival rate, i.e. $\tilde{\lambda}(t) = 1000 - \lambda(t)$ for all $t \in [0, T]$.
- *Type 4.* The system provides perfect predictions, i.e. $\tilde{\lambda}(t) = \lambda(t)$ for all $t \in [0, T]$.

The competitive ratio of both AP and ABCS decreases as more accurate predictions are available. Also, whether the system provides opposite predictions or does not provide any predictions does not matter for the competitive ratio. The competitive ratio decreases if the system predicts the average of the arrival rate. The performance of ABCS seems to increase as β decreases and θ increases, which is similar to but more pronounced as compared to the sinusoidal arrival pattern.

Confidence. Finally, we discuss the dependence of the performance on the confidence hyperparameter (r) as introduced in Corollary 4.9. Figure 9 presents the competitive ratio as a function of the confidence hyperparameter. Surprisingly, the competitive ratio decreases as the confidence increases even if the predictions are inaccurate. We attribute this to the fact that, for these arrival patterns, it is beneficial to increase the number of servers quickly in response to an increase in workload. If the confidence hyperparameter is higher and as a result R_1 is higher, then ABCS is able to increase the capacity quickly.

5.3 The offline optimum

The optimal offline solution to (2.1) can be computed by evaluating a linear program. Here, for the sake of numerical complexity, we assume that the arrival rate function is constant in each δ -interval for arbitrary $\delta > 0$ small. More specifically, we assume that there exists $\delta > 0$, such that

$$\lambda(i\delta + s) = \lambda(i\delta) \text{ for all } s \in [0, \delta) \text{ and } i = 0, 1, \dots, \lfloor T/\delta \rfloor. \quad (5.1)$$

The assumption is reasonable in practice since the smallest time unit in the datasets presented here is one second and we could take $\delta = 1$ second. Assume that T is divisible by δ and let $n = T/\delta$, $q_1 = 0$ and $m_0 = m(0)$. The linear program is

$$\begin{aligned} \underset{m, d \in \mathbb{R}^n \text{ and } q \in \mathbb{R}^{n+1}}{\text{minimize}} \quad & \omega\delta \cdot \sum_{i=1}^n \frac{q_i + q_{i+1}}{2} + \beta \cdot \sum_{i=1}^n d_i + \theta\delta \cdot \sum_{i=1}^n m_i \\ \text{subject to} \quad & q_{i+1} \geq q_i + \int_{(i-1)\delta}^{i\delta} \lambda(t) dt - \delta m_i \quad \text{for all } i = 1, \dots, n \\ & d_i \geq m_i - m_{i-1} \quad \text{for all } i = 1, \dots, n \\ & q_{i+1}, d_i, m_i \geq 0 \quad \text{for all } i = 1, \dots, n \end{aligned} \quad (5.2)$$

The linear program returns an approximation of the optimal offline solution of (2.1) as demonstrated by the next lemma. Lemma 5.1 is proved in Appendix A.6.

Lemma 5.1. *Fix a finite time horizon T , arrival rate function $\lambda(\cdot)$, and the initial number of servers $m(0)$. Let OPT be as defined in (2.3) and let $m(\cdot)$ be an optimal solution of the linear program (5.2). Then,*

$$\text{Cost}^\lambda(m, T) \leq \left(1 + \frac{\omega\delta}{2\theta}\right) \left(\left(1 + \frac{\omega\delta^2}{\beta}\right) \text{OPT} + \frac{\omega\delta^2 m(0)}{2} \right). \quad (5.3)$$

6 Proofs

6.1 Proof of Theorem 4.1

We will provide a high-level overview of the proof of Theorem 4.1 and refer to the appendix for the details. Recall that the BCS algorithm is a special case of the ABCS algorithm (let $R_1 = r_1$, $R_2 = r_2$). To prove Theorem 4.1, we will, in fact, establish a more general result in Proposition 6.1 below, where the rates r_1 and r_2 may vary as rate functions over time. Proposition 6.1 states that the competitive ratio of ABCS never exceeds PCR irrespective of the magnitude of the error in prediction. Theorem 4.1 thus follows immediately by letting $R_1 = r_1$ and $R_2 = r_2$.

Proposition 6.1. *Fix a finite time horizon T , arrival rate function $\lambda(\cdot)$ and initial number of servers $m(0)$. Let OPT be as defined in (2.3) and $m(t)$ be the number of servers of ABCS (Algorithm 3) when it has access to a prediction $\tilde{\lambda}$. Then*

$$\text{Cost}^\lambda(m, T) \leq \text{PCR} \cdot \text{OPT} + \frac{\beta \cdot m(0)}{r_2}, \quad (6.1)$$

for all instances $(T, \lambda, m(0))$ and predictions $\tilde{\lambda}$, where PCR is as defined in (4.13).

The proof of Proposition 6.1 is based on a potential function argument and is provided in Appendix A.7. We end this section by giving a proof-sketch of Proposition 6.1.

Proof sketch of Proposition 6.1. Let $m(\cdot)$ be the number of servers of ABCS and $m^*(\cdot)$ be a differentiable optimal solution to the offline optimization problem (2.1). Appendix A.7 shows how to extend this to arbitrary non-differentiable solutions. Let $\Phi(\cdot)$ be a non-negative potential function such that

$$\frac{d\Phi(t)}{dt} + \frac{\partial \text{Cost}^\lambda(m, t)}{\partial t} \leq \text{PCR} \cdot \frac{\partial \text{Cost}^\lambda(m^*, t)}{\partial t}, \quad (6.2)$$

assuming, for now, that such a $\Phi(\cdot)$ exists. We integrate equation (6.2) from time $t = 0$ to $t = T$ to obtain

$$\text{Cost}^\lambda(m, T) \leq \text{PCR} \cdot \text{Cost}^\lambda(m^*, T) + \Phi(0) - \Phi(T) \leq \text{PCR} \cdot \text{Cost}^\lambda(m^*, T) + \Phi(0), \quad (6.3)$$

where the last step follows because $\Phi(T)$ is non-negative. The proof of Proposition 6.1 is therefore completed if we manage to find a potential function $\Phi(t)$ satisfying equation (6.2) and $\Phi(0) = \frac{\beta \cdot m(0)}{r_2}$. Define the potential function $\Phi(t)$ such that

$$\begin{aligned} \Phi(t) = & \begin{cases} c_5 \beta \cdot (d_{R_1}(t) - m(t) + m^*(t)), & \text{if } m(t) > m^*(t) \\ c_6 \beta \cdot (d_{r_1}(t) - m(t) + m^*(t)) & \text{if } m(t) \leq m^*(t) \end{cases} \\ & + \frac{\beta \cdot m(t)}{r_2} + c_6 R_2 \theta \cdot [q(t) - q^*(t)]^+, \end{aligned} \quad (6.4)$$

where c_5 and c_6 are as defined in equation (4.13) and

$$d_r(t) = \sqrt{\frac{r\omega \cdot ([q(t) - q^*(t)]^+)^2}{\beta} + (m(t) - m^*(t))^2}. \quad (6.5)$$

Note that $\Phi(t)$ is non-negative and $\Phi(0) = \frac{\beta \cdot m(0)}{r_2}$. It remains to show that this choice of $\Phi(t)$ satisfies equation (6.2), which profoundly relies on $d_r(t)$. The full argument involves a case distinction and is provided in Appendix A.7. For this proof sketch, let us only consider the case that $m(t) > m^*(t)$, $q(t) > q^*(t)$, $\frac{dm}{dt} \geq 0$ and $\frac{dm^*}{dt} \geq 0$. Recall that, by the definition of ABCS,

$$\frac{dq(t)}{dt} = \lambda(t) - m(t), \quad \frac{dm(t)}{dt} = \frac{\hat{r}_1(t)\omega \cdot q(t) - \hat{r}_2(t)\theta \cdot m(t)}{\beta} \leq \frac{R_1\omega \cdot q(t)}{\beta}. \quad (6.6)$$

The derivative of $d_{R_1}(t)$ is at most

$$\begin{aligned} \beta \cdot \frac{dd_{R_1}}{dt} & \leq d_{R_1}(t)^{-1} \cdot \left(\begin{aligned} & R_1\omega \cdot (q(t) - q^*(t))(\lambda(t) - m(t)) \\ & + R_1\omega \cdot (q^*(t) - q(t))(\lambda(t) - m^*(t)) \\ & + R_1\omega \cdot q(t) \cdot (m(t) - m^*(t)) \\ & + \beta \cdot \frac{dm^*}{dt} \cdot (m^*(t) - m(t)) \end{aligned} \right) \\ & = d_{R_1}(t)^{-1} \cdot (m(t) - m^*(t)) \left(R_1\omega \cdot q^*(t) - \beta \cdot \frac{dm^*}{dt} \right) \leq R_1\omega \cdot q^*(t). \end{aligned} \quad (6.7)$$

Crucially, the derivative does not contain any terms involving $m(t)$ or $q(t)$, but only $q^*(t)$, which is easy to bound by the cost of the optimal solution. Thus, the derivative of $\Phi(t)$ can be upper bounded as follows:

$$\frac{d\Phi(t)}{dt} \leq c_5 R_1 \omega \cdot q^*(t) - \left(1 + \frac{1}{r_1}\right) \beta \cdot \frac{dm}{dt} + c_5 \beta \cdot \frac{dm^*}{dt} + \left(1 + \frac{R_2}{r_1}\right) \theta \cdot (m^*(t) - m(t)), \quad (6.8)$$

where some the constants c_5 and c_6 have been expanded according to their definitions in (4.13). Observe that the derivative of the potential function $\Phi(t)$ exactly cancels the cost of ABCS, since $\omega \cdot q(t) \leq \frac{\beta}{r_1} \cdot \frac{dm(t)}{dt} + \frac{R_2 \theta m(t)}{r_1}$. We therefore obtain

$$\begin{aligned} \frac{d\Phi(t)}{dt} + \frac{\partial \text{Cost}^\lambda(m, t)}{\partial t} &\leq c_5 R_1 \omega \cdot q^*(t) + c_5 \beta \cdot \frac{dm^*}{dt} + \left(1 + \frac{R_2}{r_1}\right) \theta \cdot m^*(t) \\ &\leq \max \left(c_5 R_1, c_5, 1 + \frac{R_2}{r_1} \right) \cdot \frac{\partial \text{Cost}^\lambda(m^*, t)}{\partial t}. \end{aligned} \quad (6.9)$$

Note that removing the term $d_r(t)$ from $\Phi(t)$ would yield a similar form as equation (6.8). However, the resulting potential function is not non-negative, hence the need for the term $d_r(t)$. \square

6.2 Proof of Theorem 4.8

Theorem 4.8 states that the competitive ratio is at most the minimum of the Optimistic Competitive Ratio (OCR) and the Pessimistic Competitive Ratio (PCR). Proposition 6.1 in the previous section showed that the competitive ratio of ABCS is at most PCR. To prove the bound on the competitive ratio by OCR, we will relate the performance of ABCS to the cost achieved by the subroutine AP. Proposition 6.2 below states that the cost of ABCS differs by at most a factor of OCR from the cost of AP.

Proposition 6.2. *Fix a finite time horizon T , arrival rate function $\lambda(\cdot)$ and initial number of servers $m(0)$. Let OPT be as defined in (2.3) and $m(t)$ be the number of servers of BCS (Algorithm 1) when it has access to a prediction $\tilde{\lambda}$. Then*

$$\text{Cost}^\lambda(m, T) \leq \text{OCR} \cdot \text{Cost}^\lambda(\tilde{m}, T) + \frac{\beta \cdot m(0)}{R_2}, \quad (6.10)$$

for all instances $(T, \lambda, m(0))$ and predictions $\tilde{\lambda}$, where $\tilde{m}(t)$ represents the advised number of servers of AP and OCR is as defined in (4.13).

The proof of Proposition 6.2 is based on a potential function argument and is provided in Appendix A.8. The proof follows a similar structure as the proof of Proposition 6.1. We now have all the ingredients to prove Theorem 4.8.

Proof of Theorem 4.8. The proof follows almost immediately from Proposition 6.1 and 6.2. Note that $\text{CR}(\eta) \leq \text{PCR}$ because

$$\text{Cost}^\lambda(m, T) \leq \text{PCR} \cdot \text{OPT} + \frac{\beta \cdot m(0)}{r_2}, \quad (6.11)$$

by Proposition 6.1. Next, $\text{CR}(\eta) \leq (1 + (\sqrt{2\omega\beta} + \theta)\eta) \cdot \text{OCR}$ because

$$\begin{aligned} \text{Cost}^\lambda(m, T) &\leq \text{OCR} \cdot \text{Cost}^\lambda(\tilde{m}, T) + \frac{\beta \cdot m(0)}{R_2} \\ &\leq (1 + (\sqrt{2\omega\beta} + \theta)\eta) \cdot \text{OCR} \cdot \text{OPT} + \frac{\beta \cdot m(0)}{R_2}, \end{aligned} \quad (6.12)$$

by Proposition 6.2 and Theorem 4.6. \square

Remark 6.3. Note that Proposition 6.2 is oblivious to the choice of AP as the source of the advised number of servers \tilde{m} . Therefore, if there exists an algorithm similar to AP but with a better error dependence, then it is straightforward to extend ABCS to use this algorithm as the source for the advised number of servers instead. As the proof of Theorem 4.8 shows, the improved error dependence carries over immediately.

7 Conclusion

In this paper, we explored how ML predictions can be used to improve the performance of capacity scaling solutions without sacrificing robustness. We extend the state of the art in capacity scaling by analyzing a more general model in continuous time where tasks are allowed to wait, in which case popular earlier proposed algorithms are not constant competitive. The Balanced Capacity Scaling (BCS) algorithm we proposed is 5-competitive in the general case. We also introduced the Adapt to the Prediction (AP) algorithm which is 1-competitive if the ML predictions are accurate. Finally, we proposed the Adaptive Balanced Capacity Scaling (ABCS) algorithm, which combines the ideas behind BCS and AP. We proved that, in the presence of accurate ML predictions, ABCS is $(1 + \varepsilon)$ -competitive and its performance degrades gracefully in the prediction's accuracy. Moreover, the competitive ratio of ABCS is at most $\Theta(\varepsilon^{-3})$ when ML predictions are inaccurate. Although the competitive ratio of ABCS depends on the accuracy, the algorithm is oblivious to it. In the context of data centers, since real-world internet traffic is erratic, any implemented capacity scaling solution must be robust against sudden unpredictable surges in workload. Our results yield significant reductions in power consumption while maintaining robustness against these sudden spikes. The theoretical results are supported by extensive numerical experiments on a real-world dataset. Finally, in an ongoing work, we are exploring how the confidence hyperparameter of ABCS can be learned over time if there are statistical guarantees on the prediction's accuracy.

8 Acknowledgements

DM thanks Ravi Kumar (Google) for inspiring discussions that started this project. We also gratefully acknowledge the financial support for this project from the ARC-TRIAD fellowship.

References

- [1] Albers, S. and Fujiwara, H. (2007). Energy-efficient algorithms for flow time minimization. *ACM Transactions on Algorithms (TALG)*, 3(4):49–es.
- [2] Albers, S., Müller, F., and Schmelzer, S. (2014). Speed scaling on parallel processors. *Algorithmica*, 68(2):404–425.
- [3] Anderson, C. and Karlin, A. R. (1996). Two adaptive hybrid cache coherency protocols. In *Proceedings. Second International Symposium on High-Performance Computer Architecture*, pages 303–313. IEEE.
- [4] Ata, B. and Shneorson, S. (2006). Dynamic control of an M/M/1 service system with adjustable arrival and service rates. *Management Science*, 52(11):1778–1791.

- [5] Augustine, J., Irani, S., and Swamy, C. (2004). Optimal power-down strategies. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 530–539. IEEE.
- [6] Azar, Y., Bartal, Y., Feuerstein, E., Fiat, A., Leonardi, S., and Rosén, A. (1999). On capital investment. *Algorithmica*, 25(1):22–36.
- [7] Bamas, E., Maggiori, A., Rohwedder, L., and Svensson, O. (2020). Learning augmented energy minimization via speed scaling. *arXiv preprint arXiv:2010.11629*.
- [8] Bansal, N., Chan, H.-L., and Pruhs, K. (2009). Speed scaling with an arbitrary power function. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 693–701. SIAM.
- [9] Barbu, V. and Precupanu, T. (2012). *Convexity and optimization in Banach spaces*. Springer Science & Business Media.
- [10] Barroso, L. A. and Hölzle, U. (2007). The case for energy-proportional computing. *Computer*, 40(12):33–37.
- [11] Bodik, P. (2010). *Automating datacenter operations using machine learning*. PhD thesis, UC Berkeley.
- [12] Boyar, J., Favrholt, L. M., Kudahl, C., Larsen, K. S., and Mikkelsen, J. W. (2017). Online algorithms with advice: A survey. *ACM Computing Surveys (CSUR)*, 50(2):1–34.
- [13] Cortez, E., Bonde, A., Muzio, A., Russinovich, M., Fontoura, M., and Bianchini, R. (2017). Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 153–167.
- [14] Damaschke, P. (2003). Nearly optimal strategies for special cases of on-line capital investment. *Theoretical Computer Science*, 302(1-3):35–44.
- [15] Dayarathna, M., Wen, Y., and Fan, R. (2015). Data center energy consumption modeling: A survey. *IEEE Communications Surveys & Tutorials*, 18(1):732–794.
- [16] Doytchinov, B., Lehoczy, J., and Shreve, S. (2001). Real-time queues in heavy traffic with earliest-deadline-first queue discipline. *Annals of Applied Probability*, pages 332–378.
- [17] Facebook (2014). Making Facebook’s software infrastructure more energy efficient with Autoscale. <https://engineering.fb.com/production-engineering/making-facebook-s-software-infrastructure-more-energy-efficient-with-autoscale/>.
- [18] Galloway, J., Smith, K., and Carver, J. (2012). An empirical study of power aware load balancing in local cloud architectures. In *2012 Ninth International Conference on Information Technology-New Generations*, pages 232–236. IEEE.
- [19] Gandhi, A., Doroudi, S., Harchol-Balter, M., and Scheller-Wolf, A. (2013). Exact analysis of the M/M/k/setup class of Markov chains via recursive renewal reward. In *Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*, pages 153–166.

- [20] Gandhi, A., Dube, P., Karve, A., Kochut, A., and Zhang, L. (2014). Adaptive, model-driven autoscaling for cloud applications. In *11th International Conference on Autonomic Computing (ICAC 14)*, pages 57–64.
- [21] Gandhi, A., Gupta, V., Harchol-Balter, M., and Kozuch, M. A. (2010). Optimality analysis of energy-performance trade-off for server farm management. *Performance Evaluation*, 67(11):1155–1171.
- [22] Gandhi, A., Harchol-Balter, M., Raghunathan, R., and Kozuch, M. A. (2012). Autoscale: Dynamic, robust capacity management for multi-tier data centers. *ACM Transactions on Computer Systems (TOCS)*, 30(4):1–26.
- [23] Gao, J. (2014). Machine learning applications for data center optimization.
- [24] Goldman, S. A., Parwatikar, J., and Suri, S. (2000). Online scheduling with hard deadlines. *Journal of Algorithms*, 34(2):370–389.
- [25] Google (2016). Data centers get fit on efficiency. <https://blog.google/outreach-initiatives/environment/data-centers-get-fit-on-efficiency/>.
- [26] Hsu, C.-Y., Indyk, P., Katabi, D., and Vakilian, A. (2019). Learning-based frequency estimation algorithms. In *International Conference on Learning Representations*.
- [27] Irani, S., Shukla, S., and Gupta, R. (2002). Competitive analysis of dynamic power management strategies for systems with multiple power saving states. In *Proceedings 2002 Design, Automation and Test in Europe Conference and Exhibition*, pages 117–123. IEEE.
- [28] Karlin, A. R., Kenyon, C., and Randall, D. (2003). Dynamic TCP acknowledgment and other stories about $e/(e-1)$. *Algorithmica*, 36:209–224.
- [29] Karlin, A. R., Manasse, M. S., Rudolph, L., and Sleator, D. D. (1988). Competitive snoopy caching. *Algorithmica*, 3(1-4):79–119.
- [30] Khanafer, A., Kodialam, M., and Puttaswamy, K. P. (2013). The constrained ski-rental problem and its application to online cloud cost optimization. In *2013 Proceedings IEEE INFOCOM*, pages 1492–1500. IEEE.
- [31] Kumar, R., Purohit, M., Schild, A., Svitkina, Z., and Vee, E. (2018). Semi-online bipartite matching. *arXiv preprint arXiv:1812.00134*.
- [32] Lassette, E., Coleman, D. W., Diao, Y., Froehlich, S., Hellerstein, J. L., Hsiung, L., Mummert, T., Raghavachari, M., Parker, G., Russell, L., et al. (2003). Dynamic surge protection: An approach to handling unexpected workload surges with resource actions that have lead times. In *International Workshop on Distributed Systems: Operations and Management*, pages 82–92. Springer.
- [33] Lee, R., Hajiesmaili, M. H., and Li, J. (2019). Learning-assisted competitive algorithms for peak-aware energy scheduling. *arXiv preprint arXiv:1911.07972*.
- [34] Lin, M., Wierman, A., Andrew, L. L., and Thereska, E. (2012). Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Transactions on Networking*, 21(5):1378–1391.

- [35] Lu, T., Chen, M., and Andrew, L. L. (2012). Simple and effective dynamic provisioning for power-proportional data centers. *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1161–1171.
- [36] Lykouris, T. and Vassilvitskii, S. (2018). Competitive caching with machine learned advice. In *International Conference on Machine Learning*, pages 3296–3305.
- [37] Maccio, V. J. and Down, D. G. (2015). On optimal policies for energy-aware servers. *Performance Evaluation*, 90:36–52.
- [38] Mahdian, M., Nazerzadeh, H., and Saberi, A. (2012). Online optimization with uncertain information. *ACM Transactions on Algorithms (TALG)*, 8(1):1–29.
- [39] Manmeet, S., Maninder, S., and Sanmeet, K. (2019). TI-2016 DNS dataset. *IEEE Dataport*.
- [40] Mazzucco, M. and Dyachuk, D. (2012). Optimizing cloud providers revenues via energy efficient server allocation. *Sustainable Computing: Informatics and Systems*, 2(1):1–12.
- [41] Mitzenmacher, M. (2018). A model for learned bloom filters and optimizing by sandwiching. In *Advances in Neural Information Processing Systems*, pages 464–473.
- [42] Mitzenmacher, M. (2019a). Scheduling with predictions and the price of misprediction. *arXiv preprint arXiv:1902.00732*.
- [43] Mitzenmacher, M. (2019b). The supermarket model with known and predicted service times. *arXiv preprint arXiv:1905.12155*.
- [44] Mukherjee, D., Dhara, S., Borst, S. C., and van Leeuwen, J. S. H. (2017). Optimal service elasticity in large-scale distributed systems. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(1):1–28.
- [45] Mukherjee, D. and Stolyar, A. (2019). Join idle queue with service elasticity: Large-scale asymptotics of a nonmonotone system. *Stochastic Systems*, 9(4):338–358.
- [46] Nature (2018). How to stop data centres from gobbling up the world’s electricity. <https://www.nature.com/articles/d41586-018-06610-y>.
- [47] Netflix (2013). Scryer: Netflix’s Predictive Auto Scaling Engine. <https://netflixtechblog.com/scryer-netflixs-predictive-auto-scaling-engine-a3f8fc922270>.
- [48] Purohit, M., Svitkina, Z., and Kumar, R. (2018). Improving online algorithms via ml predictions. In *Advances in Neural Information Processing Systems*, pages 9661–9670.
- [49] Qi, J., Du, J., Siniscalchi, S. M., Ma, X., and Lee, C.-H. (2020). On mean absolute error for deep neural network based vector-to-vector regression. *IEEE Signal Processing Letters*.
- [50] Rong, H., Zhang, H., Xiao, S., Li, C., and Hu, C. (2016). Optimizing energy consumption for data centers. *Renewable and Sustainable Energy Reviews*, 58:674–691.
- [51] Rządca, K., Findeisen, P., Swiderski, J., Zych, P., Broniek, P., Kusmierek, J., Nowak, P., Strack, B., Witusowski, P., Hand, S., et al. (2020). Autopilot: workload autoscaling at google. In *Proceedings of the Fifteenth European Conference on Computer Systems*, pages 1–16.

- [52] Shehabi, A., Smith, S., Sartor, D., Brown, R., Herrlin, M., Koomey, J., Masanet, E., Horner, N., Azevedo, I., and Lintner, W. (2016). United States Data Center Energy Usage Report. Technical report, Lawrence Berkeley National Lab.
- [53] Sverdlik, Y. (2020). How Zoom, Netflix, and Dropbox are Staying Online During the Pandemic. <https://www.datacenterknowledge.com/uptime/how-zoom-netflix-and-dropbox-are-staying-online-during-pandemic>.
- [54] Tirmazi, M., Barker, A., Deng, N., Haque, M. E., Qin, Z. G., Hand, S., Harchol-Balter, M., and Wilkes, J. (2020). Borg: the next generation. In *Proceedings of the Fifteenth European Conference on Computer Systems*, pages 1–14.
- [55] Urgaonkar, B., Shenoy, P., Chandra, A., and Goyal, P. (2005). Dynamic provisioning of multi-tier internet applications. In *Second International Conference on Autonomic Computing (ICAC'05)*, pages 217–228. IEEE.
- [56] Wierman, A., Andrew, L. L., and Tang, A. (2009). Power-aware speed scaling in processor sharing systems. In *IEEE INFOCOM 2009*, pages 2007–2015. IEEE.

A Proofs

This section provides the proofs which have been omitted from the main text.

A.1 Proof of Proposition 2.2

Fix a finite time horizon T , arrival rate function $\lambda(\cdot)$ and initial number of servers $m(0)$. Let

$$V^+(f) = \limsup_{\delta \downarrow 0} \sum_{i=0}^{\lfloor T/\delta \rfloor} [f(i\delta + \delta) - f(i\delta)]^+ \quad (\text{A.1})$$

for a function $f : [0, T] \rightarrow \mathbb{R}$. The definition of $V^+(f)$ is closely related to the notion of bounded variation. The bounded variation of a function $f : [0, T] \rightarrow \mathbb{R}$ is defined as

$$V(f) = \sup \left\{ \sum_{i=1}^n |f(z_i) - f(z_{i-1})| \text{ such that } \{z_i\}_{i=0}^n \text{ is a partition of } [0, T] \right\}. \quad (\text{A.2})$$

Let m_n be a sequence of functions such that $\text{Cost}^\lambda(m_n, T) \rightarrow \text{OPT}$ as $n \rightarrow \infty$. There exists $N \in \mathbb{N}$ such that $\text{Cost}^\lambda(m_n, T) \leq 2 \cdot \text{OPT}$ for all $n \geq N$. As a result, $V^+(m_n)$ is uniformly bounded for $n \geq N$. Note that, without increasing the cost, we can set $m_n(T) = 0$. The bounded variation and $V^+(m_n)$ are then related as

$$V(m_n) = 2V^+(m_n) + m(0). \quad (\text{A.3})$$

The rest of the proof depends on the following compactness theorem [9].¹

Theorem A.1. (*Helly's selection theorem*) Let $f_n : [0, T] \rightarrow \mathbb{R}$ be a sequence of functions and suppose that the next two conditions hold:

- (i) The sequence f_n has uniformly bounded variation, i.e., $\sup_{n \in \mathbb{N}} V(f_n) < \infty$,

¹DM: Can you cite precise theorem number?

(ii) The sequence f_n is uniformly bounded at a point, i.e., there exists $t \in [0, T]$ such that $\{f_n(t)\}_{n=1}^\infty$ is a bounded set.

Then, there exists a subsequence f_{n_k} of f_n and a function $f : [0, T] \rightarrow \mathbb{R}$ such that

(i) f_{n_k} converges to f pointwise as $k \rightarrow \infty$,

(ii) f_{n_k} converges to f in L_1 as $k \rightarrow \infty$,

(iii) $V(f) \leq \liminf_{k \rightarrow \infty} V(f_{n_k})$.

Recall the infinite sequence m_N, m_{N+1}, \dots introduced above. Condition (i) in Theorem A.1 holds because

$$V(m_n) = 2V^+(m_n) + m(0) \leq \frac{4\text{OPT}}{\beta} + m(0), \quad (\text{A.4})$$

for all $n \geq N$. Moreover, condition (ii) in Theorem A.1 holds for $t = 0$, since $m_n(0) = m(0)$ for all $n \in \mathbb{N}$. Hence, there exists a subsequence m_{n_k} of m_n and a function $m^* : [0, T] \rightarrow \mathbb{R}$ such that $m_{n_k} \rightarrow m^*$ pointwise and in the L_1 norm, as $k \rightarrow \infty$, and

$$V^+(m^*) = (V(m^*) - m(0))/2 \leq \liminf_{k \rightarrow \infty} (V(m_{n_k}) - m(0))/2 = \liminf_{k \rightarrow \infty} V^+(m_{n_k}). \quad (\text{A.5})$$

Therefore, since the flow-time and the power cost are continuous in m with respect to the L_1 norm,

$$\begin{aligned} \text{Cost}^\lambda(m^*, T) &= \omega \cdot \int_0^T q^*(t) dt + \beta \cdot V^+(m^*) + \theta \cdot \int_0^T m^*(t) dt \\ &\leq \omega \cdot \lim_{k \rightarrow \infty} \int_0^T q_{n_k}(t) dt + \beta \cdot \liminf_{k \rightarrow \infty} V^+(m_{n_k}) + \theta \cdot \lim_{k \rightarrow \infty} \int_0^T m_{n_k}(t) dt \\ &\leq \lim_{k \rightarrow \infty} \text{Cost}^\lambda(m_{n_k}, T) = \text{OPT}, \end{aligned} \quad (\text{A.6})$$

which completes the proof of the proposition.

A.2 Proof of Lemma 2.4

Fix any algorithm \mathcal{A} and parameters ω and β . We will construct an instance for which $\text{Cost}^\lambda(m, T) \geq \frac{\omega\tau}{2\beta} \cdot \text{OPT} + \Phi(0)$. Throughout the example we will assume that $\Phi(0)$ is zero without loss of generality.

Let $\lambda(t) = \rho\delta_\tau(t)$, where $\delta_\tau(t)$ is the Dirac delta function at τ , i.e., $\delta_\tau(\tau) = \infty$, $\delta_\tau(t) = 0$ for all $t \neq \tau$ and $\int_0^\infty \delta_\tau(s) ds = 1$. The value of ρ will be specified later. Let $m(0) = 0$, the finite time horizon $T = 2\tau$ and $\theta = 0$. Let $m(t)$ be the number of servers of \mathcal{A} for the instance. We distinguish two cases depending on $\sup_{t \in [\tau, 2\tau)} m(t)$.

1. First, consider the case when $\sup_{t \in [\tau, 2\tau)} m(t) < \tau^{-1}$. Fix $\rho = 2$. One possible alternative solution of (2.1) is $m^*(t) = 2$ for $t \in [\tau, 2\tau]$. This solution does not incur any waiting cost. The value of the optimal solution is therefore at most $\text{OPT} \leq 2\beta$. The cost of \mathcal{A} is at least $\text{Cost}^\lambda(m, T) > \omega \cdot \int_\tau^{2\tau} (\rho - (s - \tau)\tau^{-1}) ds \geq \omega\tau \geq \frac{\omega\tau}{2\beta} \cdot \text{OPT}$.
2. Next, consider the case when $\sup_{t \in [\tau, 2\tau)} m(t) \geq \tau^{-1}$. Fix $\rho = 0$. The optimal solution of (2.1) is $m^*(t) = 0$ for $t \in [0, T]$. The value of the optimal solution is $\text{OPT} = 0$. The cost of \mathcal{A} is at least $\text{Cost}^\lambda(m, T) \geq \beta \cdot \tau^{-1} \geq \frac{\omega\tau}{2\beta} \cdot \text{OPT}$.

This completes the proof of the Lemma.

A.3 Proof of Theorem 4.3

Fix a finite time horizon T , arrival rate function $\lambda(\cdot)$ and initial number of servers $m(0)$. Let $m^*(t)$ be a solution of the offline optimization problem (2.1) and $m(t)$ be the number of servers of BCS (Algorithm 1). We will prove that

$$\text{Cost}(m, T) \leq 2 \cdot \text{Cost}(m^*, T) + \beta \cdot m(0), \quad (\text{A.7})$$

where we have omitted λ from the notation $\text{Cost}^\lambda(m, T)$.

Overview of the proof. Let $t_1 \leq t_2 \leq \dots$ be a partitioning of the interval $[0, T]$ such that (i) $m(t)$ is monotone in $[t_k, t_{k+1}]$ and (ii) either $m(t) > m^*(t)$ or $m(t) \leq m^*(t)$ in $[t_k, t_{k+1}]$ for all $k \in \mathbb{N}$. The goal of the proof will be to find a non-negative potential function $\Phi(t)$ such that

$$\Phi(t_{k+1}) - \Phi(t_k) + \text{Cost}(m, t_{k+1}) - \text{Cost}(m, t_k) \leq 2 \cdot (\text{Cost}(m^*, t_{k+1}) - \text{Cost}(m^*, t_k)), \quad (\text{A.8})$$

for all $k \in \mathbb{N}$. We sum equation (A.8) over $k \in \mathbb{N}$ to obtain

$$\text{Cost}(m, T) \leq 2 \cdot \text{Cost}(m^*, T) + \Phi(0) - \Phi(T) \leq 2 \cdot \text{Cost}(m^*, T) + \Phi(0), \quad (\text{A.9})$$

where the last step follows because $\Phi(T)$ is non-negative. The proof of Theorem 4.3 is therefore completed if we manage to find a non-negative potential function $\Phi(t)$ satisfying equation (A.8) and $\Phi(0) = \beta \cdot m(0)$.

Choice of $\Phi(t)$. Define the potential function $\Phi(t)$ such that

$$\Phi(t) = \begin{cases} \beta \cdot m(t), & \text{if } m(t) > m^*(t) \\ 2\beta \cdot m^*(t) - \beta \cdot m(t) & \text{if } m(t) \leq m^*(t) \end{cases} \quad (\text{A.10})$$

Note that $\Phi(t)$ is non-negative and $\Phi(0) = \beta \cdot m(0)$.

Verification of (A.8). We continue by verifying equation (A.8). Fix $k \in \mathbb{N}$. We distinguish two cases, depending on whether $m(t)$ is decreasing or non-decreasing in $[t_k, t_{k+1}]$.

(i) Assume that $m(s)$ is decreasing for $s \in [t_k, t_{k+1}]$. Recall that, by definition,

$$\frac{dm(t)}{dt} = -\frac{\theta \cdot m(t)}{\beta}, \quad (\text{A.11})$$

for $t \in [t_k, t_{k+1}]$ and therefore

$$m(t_k + s) = m(t_k) \cdot \exp\left(-\frac{\theta \cdot s}{\beta}\right), \quad (\text{A.12})$$

for $s \in [0, t_{k+1} - t_k]$ and hence

$$\theta \cdot \int_{t_k}^{t_{k+1}} m(s) ds = \beta \cdot m(t_k) \left(1 - \exp\left(-\frac{\theta \cdot (t_{k+1} - t_k)}{\beta}\right)\right) = \beta \cdot (m(t_k) - m(t_{k+1})). \quad (\text{A.13})$$

We further distinguish two cases depending on whether $m(t) > m^*(t)$ or $m(t) \leq m^*(t)$ in $[t_k, t_{k+1}]$. First, consider the case that $m(s) > m^*(s)$ for $s \in [t_k, t_{k+1}]$. Then,

$$\begin{aligned} & \Phi(t_{k+1}) - \Phi(t_k) + \text{Cost}(m, t_{k+1}) - \text{Cost}(m, t_k) \\ &= \beta \cdot (m(t_{k+1}) - m(t_k)) + \beta \cdot (m(t_k) - m(t_{k+1})) \\ &= 0 \leq 2(\text{Cost}(m^*, t_{k+1}) - \text{Cost}(m^*, t_k)). \end{aligned} \quad (\text{A.14})$$

Next, consider the case that $m(s) \leq m^*(s)$ for $s \in [t_k, t_{k+1}]$. Then,

$$\begin{aligned} & \Phi(t_{k+1}) - \Phi(t_k) + \text{Cost}(m, t_{k+1}) - \text{Cost}(m, t_k) \\ &= 2\beta \cdot (m^*(t_{k+1}) - m^*(t_k)) - \beta \cdot (m(t_{k+1}) - m(t_k)) + \beta \cdot (m(t_k) - m(t_{k+1})) \\ &= 2\beta \cdot (m^*(t_{k+1}) - m^*(t_k)) + 2\theta \cdot \int_{t_k}^{t_{k+1}} m(s) \, ds \\ &\leq 2\beta \cdot (m^*(t_{k+1}) - m^*(t_k)) + 2\theta \cdot \int_{t_k}^{t_{k+1}} m^*(s) \, ds \\ &\leq 2(\text{Cost}(m^*, t_{k+1}) - \text{Cost}(m^*, t_k)). \end{aligned} \quad (\text{A.15})$$

- (ii) Assume that $m(s)$ is non-decreasing for $s \in [t_k, t_{k+1}]$. Note that, since tasks are not allowed to wait, $m^*(t) \geq \lambda(t)$ for all $t \in [0, T]$. Recall that, by definition, if the arrival rate $\lambda(t)$ is higher than the number of servers $m(t)$ then BCS increases the number of servers to match the arrival rate. Therefore, $m(s) = \lambda(s)$ for $s \in [t_k, t_{k+1}]$ because $m(t)$ is non-decreasing in $[t_k, t_{k+1}]$. Hence, $m^*(s) \geq m(s) = \lambda(s)$ for $s \in [t_k, t_{k+1}]$ and

$$\begin{aligned} & \Phi(t_{k+1}) - \Phi(t_k) + \text{Cost}(m, t_{k+1}) - \text{Cost}(m, t_k) \\ &= 2\beta \cdot (m^*(t_{k+1}) - m^*(t_k)) - \beta \cdot (m(t_{k+1}) - m(t_k)) \\ &\quad + \beta \cdot (m(t_{k+1}) - m(t_k)) + \theta \cdot \int_{t_k}^{t_{k+1}} m(s) \, ds \\ &\leq 2\beta \cdot (m^*(t_{k+1}) - m^*(t_k)) + \theta \cdot \int_{t_k}^{t_{k+1}} m^*(s) \, ds \\ &\leq 2(\text{Cost}(m^*, t_{k+1}) - \text{Cost}(m^*, t_k)). \end{aligned} \quad (\text{A.16})$$

A.4 Proof of Proposition 4.4

Fix any algorithm \mathcal{A} , and let $m(t)$ denote its number of servers. We will construct an instance $(T, \lambda, m(0))$ for which $\text{Cost}^\lambda(m, T) \geq 2.549 \cdot \text{OPT} + \Phi(0)$. Throughout the example, we will assume that $\Phi(0)$ is zero, without loss of generality.

Let $\lambda(t) = 1$ for $t \in [0, T]$ and $m(0) = 0$. The time horizon T will be specified later. Fix $\beta = \omega = 1$ and $\theta = 0$. Let the prediction $\tilde{\lambda}(t) = 0$ for all t and as a result the advised number of servers $\tilde{m}(t) = 0$ for all t . Let $m(t)$ be the number of servers of \mathcal{A} for the instance. Define $\tau := \inf\{t \mid m(t) > 0.885t^2\}$ or $\tau = \infty$, if the infimum does not exist. We distinguish two cases depending on the value of τ .

1. First, consider the case when $\tau \leq 1.225$. Fix $T = \tau$. The optimal solution to (2.1) is $m^*(t) = 0$ for $t \in [0, T]$. The value of the optimal solution is purely due to flow-time and is equal to $\text{OPT} = \tau^2/2$.

At time $t = \tau$, algorithm \mathcal{A} has at least $m(\tau) > 0.885\tau^2$ servers. The flow-time is at least $\int_0^\tau q(t) \, dt \geq \int_0^\tau \int_0^t 1 - 0.885s^2 \, ds \, dt \geq \tau^2/2 - 0.885\tau^4/12$, because $m(t) \leq 0.885t^2$ for

$t \in [0, \tau)$. The cost of \mathcal{A} is therefore at least $\text{Cost}^\lambda(m, T) \geq 0.885\tau^2 + \tau^2/2 - 0.885\tau^4/12 \geq 2.549 \cdot \tau^2/2 = 2.549 \cdot \text{OPT}$, where the second inequality follows because $\tau \leq 1.225$.

2. Next, consider the case when $\tau > 1.225$. Fix $T = 3$. The optimal solution to (2.1) is $m^*(t) = 1$ for $t \in [0, T]$. The value of the optimal solution is purely due to switching cost and is equal to $\text{OPT} = 1$.

At time $t = 1.225$, the queue length of \mathcal{A} is at least $q(1.225) \geq \int_0^{1.225} 1 - 0.885t^2 dt = 0.682$. The optimal solution starting from time $t = 1.225$ is $m(t) = 1 + q(1.225)/\sqrt{2} \geq 1.483$ for $t \in (1.225, T]$. The flow-time is therefore at least $\int_0^T q(t) dt \geq \int_0^{1.225} \int_0^s 1 - 0.885s^2 ds dt + q(1.225)/\sqrt{2} \geq 1.067$, again because $m(t) \leq 0.885t^2$ for $t \in [0, \tau)$. The cost of \mathcal{A} is therefore at least $\text{Cost}^\lambda(m, T) \geq 2.549 \geq 2.549 \cdot \text{OPT}$.

Hence, the statement follows.

A.5 Proof of Theorem 4.6

Proof. Fix a finite time horizon T , arrival rate function $\lambda(\cdot)$ and initial number of servers $m(0)$. Let $\tilde{\lambda}(\cdot)$ be the predicted arrival rate. The idea to the proof is to separate the cost into the cost of the offline and the online component. More specifically, we claim that

$$\text{Cost}^\lambda(m, T) \leq \text{Cost}^{\tilde{\lambda}}(m_1, T) + \left(\sqrt{2\omega\beta} + \theta \right) T \cdot \|\Delta\lambda\|_{MAE}. \quad (\text{A.17})$$

To see why, note that

$$\begin{aligned} q(t) &= \int_0^t (\lambda(s) - m(s)) \mathbb{1}\{q(s) > 0 \text{ or } \lambda(s) \geq m(s)\} ds \\ &\leq \int_0^t (\tilde{\lambda}(s) - m_1(s)) \mathbb{1}\{q(s) > 0 \text{ or } \lambda(s) \geq m(s)\} ds \\ &\quad + \int_0^t (\Delta\lambda(s) - m_2(s)) \mathbb{1}\{q(s) > 0 \text{ or } \lambda(s) \geq m(s)\} ds \\ &\leq \int_0^t (\tilde{\lambda}(s) - m_1(s)) \mathbb{1}\{q_1(s) > 0 \text{ or } \lambda(s) \geq m_1(s)\} ds \\ &\quad + \int_0^t (\Delta\lambda(s) - m_2(s)) \mathbb{1}\{q_2(s) > 0 \text{ or } \lambda(s) \geq m_2(s)\} ds \\ &= q_1(t) + q_2(t). \end{aligned} \quad (\text{A.18})$$

Therefore, the flow-time of the algorithm is at most the sum of the flow-time of the offline component on $\tilde{\lambda}$ and the online component on $\Delta\lambda$. Similarly, since the switching cost and the power cost are linear in the number of servers $m(\cdot)$, the cost of the algorithm is at most

$$\text{Cost}^\lambda(m, T) \leq \text{Cost}^{\tilde{\lambda}}(m_1, T) + \text{Cost}^{\Delta\lambda}(m_2, T). \quad (\text{A.19})$$

We will further bound the cost of the online component m_2 . Let $[t, t + \delta) \subseteq [0, T]$ be an arbitrary time interval for $\delta > 0$ small and let $\Delta q(t) = \int_t^{t+\delta} \Delta\lambda(s) ds$. We will bound the cost due to the $\Delta q(t)$ workload received in this time interval. The number of servers $m_2(t)$ increases by $\sqrt{\omega/(2\beta)} \cdot \Delta q(t)$ in the interval. Moreover, after a time of $\sqrt{2\beta/\omega}$, the number of servers $m_2(t)$ decreases again by $\sqrt{\omega/(2\beta)} \cdot \Delta q(t)$. Throughout $[t, t + \sqrt{2\beta/\omega})$, the queue length due to this

fraction of the workload decreases linearly as $q(t+s) = \Delta q(t) - \sqrt{\omega/(2\beta)} \cdot \Delta q(t) \cdot s$ until the workload is completely handled. The cost due to waiting is therefore

$$\omega \cdot \int_0^{\sqrt{\frac{2\beta}{\omega}}} \left(\Delta q(t) - \sqrt{\frac{\omega}{2\beta}} \cdot \Delta q(t) \cdot s \right) ds = \omega \cdot \sqrt{\frac{\beta}{2\omega}} \cdot \Delta q(t). \quad (\text{A.20})$$

Note that, since δ can be chosen arbitrarily small, the waiting cost in the interval $[t, t+\delta)$ is negligible. The switching cost is $\beta \cdot \sqrt{\omega/(2\beta)} \cdot \Delta q(t)$ and the power cost is $\theta \cdot \sqrt{2\beta/\omega} \cdot \sqrt{\omega/(2\beta)} \cdot \Delta q(t) = \theta \cdot \Delta q(t)$. The cost of the online component is therefore

$$\text{Cost}^{\Delta\lambda}(m_2, T) \leq \lim_{\delta \downarrow 0} \sum_{i=0}^{\lfloor T/\delta \rfloor} \left(\sqrt{2\omega\beta} + \theta \right) \cdot \Delta q(i\delta) = \left(\sqrt{2\omega\beta} + \theta \right) T \cdot \|\Delta\lambda\|_{MAE}, \quad (\text{A.21})$$

which proves equation (A.17) by combining (A.19) and (A.21). Similarly, let $\Delta\lambda^*(t) = \tilde{\lambda}(t) - \lambda(t)$. Then, by interchanging the actual arrival rate λ and the predicted arrival rate $\tilde{\lambda}$ in equation (A.17), we find that

$$\text{Cost}^{\tilde{\lambda}}(m^*, T) \leq \text{Cost}^{\lambda}(m^{*1}, T) + \text{Cost}^{\Delta\lambda^*}(m^{*2}, T), \quad (\text{A.22})$$

where $m^*(t) = m^{*1}(t) + m^{*2}(t)$ and

$$m^{*1} \in \arg \min_{m: (0, T] \rightarrow \mathbb{R}_+} \text{Cost}^{\lambda}(m, T), \quad (\text{A.23})$$

$$\frac{dm^{*2}(t)}{dt} = \sqrt{\frac{\omega}{2\beta}} \cdot \left(\Delta\lambda^*(t) - \Delta\lambda^*\left(t - \sqrt{2\beta/\omega}\right) \right). \quad (\text{A.24})$$

Finally, we combine (A.17) and (A.22) to find that

$$\begin{aligned} \text{Cost}^{\lambda}(m, T) &\leq \text{Cost}^{\tilde{\lambda}}(m_1, T) + \left(\sqrt{2\omega\beta} + \theta \right) T \cdot \|\Delta\lambda\|_{MAE} \\ &\leq \text{Cost}^{\tilde{\lambda}}(m^*, T) + \left(\sqrt{2\omega\beta} + \theta \right) T \cdot \|\Delta\lambda\|_{MAE} \\ &\leq \text{Cost}^{\lambda}(m^{*1}, T) + \left(\sqrt{2\omega\beta} + \theta \right) T \cdot \|\tilde{\lambda} - \lambda\|_{MAE}, \end{aligned} \quad (\text{A.25})$$

where the first inequality follows by (A.17), the second inequality follows because m_1 achieves the minimum cost on $\tilde{\lambda}$ and the third inequality follows by (A.22). This completes the proof because m^{*1} is the optimal offline solution on λ . \square

A.6 Proof of Lemma 5.1

Fix a finite time horizon T , arrival rate function $\lambda(\cdot)$ and initial number of servers $m(0)$. Assume that T is divisible by δ and let $n = T/\delta$. Let

$$\mathcal{C} := \{f : [0, T] \rightarrow \mathbb{R}_+ \mid f(i\delta + s) = f(i\delta) \text{ for all } s \in [0, \delta) \text{ and } i = 0, 1, \dots, n-1\} \quad (\text{A.26})$$

be the subspace of the space of functions which are constant in each δ -interval. Recall that by assumption, $\lambda \in \mathcal{C}$. We note that each $f \in \mathcal{C}$ is equivalently represented by a vector $f = (f(0), f(\delta), \dots, f(n-1)) \in \mathbb{R}^n$ and vice versa. We will therefore interchangeably use vector notation to denote an element from \mathcal{C} .

Claim A.2. *We claim that*

$$\inf_{m \in \mathcal{C}} \text{Cost}^{\lambda}(m, T) \leq \left(1 + \frac{\omega\delta^2}{\beta} \right) \inf_{m: [0, T] \rightarrow \mathbb{R}_+} \text{Cost}^{\lambda}(m, T) + \frac{\omega\delta^2 \cdot m(0)}{2}. \quad (\text{A.27})$$

Proof. Let $m^* : [0, T] \rightarrow \mathbb{R}_+$ be arbitrary and let $m_i = \frac{1}{\delta} \int_{(i-1)\delta}^{i\delta} m^*(t) dt$. We will prove that

$$\text{Cost}^\lambda(m, T) \leq \left(1 + \frac{\omega\delta^2}{\beta}\right) \text{Cost}^\lambda(m^*, T) + \frac{\omega\delta^2 \cdot m(0)}{2}, \quad (\text{A.28})$$

which finishes the proof of the claim. Note that it follows immediately by construction that the switching cost of m is at most the switching cost of m^* and the power cost of m is equal to the power cost of m^* . We will therefore focus on the flow-time cost. The queue length of m^* at the endpoints of each δ -interval is at least the queue length of m as follows,

$$\begin{aligned} q^*(i\delta) &= q^*((i-1)\delta) + \int_{(i-1)\delta}^{i\delta} (\lambda_i - m^*(s)) \mathbb{1}\{q^*(s) > 0 \text{ or } \lambda_i \geq m^*(s)\} ds \\ &\geq \left[q^*((i-1)\delta) + \int_{(i-1)\delta}^{i\delta} (\lambda_i - m^*(s)) ds \right]^+ \\ &\geq [q((i-1)\delta) + \delta\lambda_i - \delta m_i]^+ = q(i\delta), \end{aligned} \quad (\text{A.29})$$

where the inequality $q^*((i-1)\delta) \geq q((i-1)\delta)$ follows by induction on i . Define

$$\Delta_i = \sup_{t \in [(i-1)\delta, i\delta]} m^*(t) - \inf_{t \in [(i-1)\delta, i\delta]} m^*(t), \quad (\text{A.30})$$

and observe that

$$\sum_{i=1}^n \Delta_i \leq m(0) + 2 \lim_{\varepsilon \downarrow 0} \sum_{i=0}^{\lfloor T/\varepsilon \rfloor} [m^*(i\varepsilon + \varepsilon) - m^*(i\varepsilon)]^+ \leq m(0) + \frac{2\text{Cost}^\lambda(m^*, T)}{\beta}. \quad (\text{A.31})$$

Then, the flow-time cost of m^* in each δ -interval is at least,

$$\begin{aligned} \int_{(i-1)\delta}^{i\delta} q^*(t) dt &= \int_{(i-1)\delta}^{i\delta} \left[q^*((i-1)\delta) + \int_{(i-1)\delta}^t (\lambda_i - m(s)) \mathbb{1}\{q^*(s) > 0 \text{ or } \lambda_i \geq m(s)\} ds \right] dt \\ &\geq \int_{(i-1)\delta}^{i\delta} \left[q^*((i-1)\delta) + \int_{(i-1)\delta}^t (\lambda_i - m(s)) ds \right]^+ dt \\ &\geq \int_{(i-1)\delta}^{i\delta} \left[q((i-1)\delta) + \int_{(i-1)\delta}^t (\lambda_i - m_i - \Delta_i) ds \right]^+ dt \\ &= \int_{(i-1)\delta}^{i\delta} [q(t) - (t - (i-1)\delta)\Delta_i]^+ dt \geq \int_{(i-1)\delta}^{i\delta} q(t) dt - \frac{\delta^2 \Delta_i}{2}, \end{aligned} \quad (\text{A.32})$$

where the second inequality uses (A.29). Therefore,

$$\omega \cdot \int_0^T q(t) dt - \omega \cdot \int_0^T q^*(t) dt \leq \frac{\omega\delta^2}{2} \cdot \sum_{i=1}^n \Delta_i \leq \frac{\omega\delta^2 \cdot m(0)}{2} + \frac{\omega\delta^2 \cdot \text{Cost}^\lambda(m^*, T)}{\beta}, \quad (\text{A.33})$$

where the second inequality follows by (A.31). This completes the proof of the claim. \square

Let $\text{Obj}^\lambda(m, T)$ denote the value of the objective in (5.2) for $m \in \mathcal{C}$.

Claim A.3. *We claim that*

$$\begin{aligned} \text{Obj}^\lambda(m, T) &= \text{Cost}^\lambda(m, T) + \omega \cdot \sum_{i=1}^n \left(\delta \cdot \frac{q_i}{2} - \frac{q_i^2}{2(m_i - \lambda_i)} \right) \mathbb{1}\{q_i > 0 \text{ and } q_{i+1} = 0\} \\ &\leq \left(1 + \frac{\omega\delta}{2\theta}\right) \text{Cost}^\lambda(m, T), \end{aligned} \quad (\text{A.34})$$

for any $m \in \mathcal{C}$.

Proof. Let $m \in \mathcal{C}$ be arbitrary. Note that, since $m \in \mathcal{C}$, the switching cost is $\sum_{i=1}^n [m_i - m_{i-1}]^+$ and the power cost is $\sum_{i=1}^n \delta m_i$, which matches the terms in $\text{OBJ}^\lambda(m, T)$. We will therefore focus on the flow-time cost. Denote $q = (q(0), q(\delta), \dots, q(n)) \in \mathbb{R}^{n+1}$. The flow-time is equal to

$$\begin{aligned} \int_{(i-1)\delta}^{i\delta} q(t) dt &= \int_0^\delta [q_i + (\lambda_i - m_i)t]^+ dt \\ &= \delta \cdot \frac{q_i + q_{i+1}}{2} \mathbb{1}\{q_i = 0 \text{ or } q_{i+1} > 0\} + \frac{q_i^2}{2(m_i - \lambda_i)} \mathbb{1}\{q_i > 0 \text{ and } q_{i+1} = 0\}, \end{aligned} \quad (\text{A.35})$$

because $q(\cdot)$ increases or decreases linearly. This completes the equality in the claim. To see why the inequality holds, note that

$$\begin{aligned} \sum_{i=1}^n \left(\delta \cdot \frac{q_i}{2} - \frac{q_i^2}{2(m_i - \lambda_i)} \right) \mathbb{1}\{q_i > 0 \text{ and } q_{i+1} = 0\} &\leq \frac{\delta}{2} \cdot \sum_{i=1}^n q_i \mathbb{1}\{q_i > 0 \text{ and } q_{i+1} = 0\} \\ &\leq \frac{\delta}{2} \cdot \sum_{i=1}^n \delta m_i \leq \frac{\delta \cdot \text{Cost}^\lambda(m, T)}{2\theta}, \end{aligned} \quad (\text{A.36})$$

where the second inequality follows because $\delta(m_i - \lambda_i) \geq q_i$. This completes the proof of the claim. \square

We now finish the proof of Lemma 5.1. Let $m \in \mathcal{C}$ be an optimal solution to (5.2). Moreover, define

$$m^* = \arg \min_{m \in \mathcal{C}} \text{Cost}^\lambda(m, T). \quad (\text{A.37})$$

Then, the cost of m is at most,

$$\begin{aligned} \text{Cost}^\lambda(m, T) &\leq \text{OBJ}^\lambda(m, T) \leq \text{OBJ}^\lambda(m^*, T) \\ &\leq \left(1 + \frac{\omega\delta}{2\theta}\right) \text{Cost}^\lambda(m^*, T) \\ &\leq \left(1 + \frac{\omega\delta}{2\theta}\right) \left(\left(1 + \frac{\omega\delta^2}{\beta}\right) \text{OPT} + \frac{\omega\delta^2 m(0)}{2} \right), \end{aligned} \quad (\text{A.38})$$

which completes the proof of the lemma.

A.7 Proof of Proposition 6.1

Fix a finite time horizon T , arrival rate function $\lambda(\cdot)$ and initial number of servers $m(0)$. Let $m^*(t)$ be a solution of the offline optimization problem (2.1) and $q^*(t)$ the corresponding workload. Assume that the solution achieves a finite cost. If there does not exist a solution which achieves finite cost, then Proposition 6.1 follows immediately. Without loss of generality, assume that $m^*(t)$ is differentiable. To see why this is possible, assume that $m^*(t)$ is not differentiable. Define the interpolation $m_\delta^*(t)$ of $m^*(t)$ such that

$$m_\delta^*(t) = \int_t^{t+\delta} \frac{m^*(s)}{\delta} ds, \quad (\text{A.39})$$

which is differentiable for all $\delta > 0$. Also, note that

$$\int_{t_1}^{t_2} m_\delta^*(t) dt = \int_{t_1}^{t_2} \int_t^{t+\delta} \frac{m^*(s)}{\delta} ds dt \rightarrow \int_{t_1}^{t_2} m^*(t) dt \text{ as } \delta \rightarrow 0, \quad (\text{A.40})$$

for any $0 \leq t_1 \leq t_2 \leq \infty$. The cost of $m^*(t)$ and $m_\delta^*(t)$ therefore coincide, asymptotically as $\delta \rightarrow 0$. As a result, each function $m^*(\cdot)$ can be written as the limit of a sequence of differentiable functions $m_\delta^*(\cdot)$ and we therefore assume that $m^*(t)$ is differentiable without loss of generality.

Overview of the proof. Let $m(t)$ be the number of servers of ABCS (Algorithm 3) and $q(t)$ be the corresponding workload. The goal of the proof will be to find a non-negative potential function $\Phi(t)$ such that

$$\frac{d\Phi(t)}{dt} + \frac{\partial \text{Cost}(m, t)}{\partial t} \leq \text{PCR} \cdot \frac{\partial \text{Cost}(m^*, t)}{\partial t}, \quad (\text{A.41})$$

where we have omitted λ from the notation $\text{Cost}^\lambda(m, t)$. Note that $\text{Cost}(m, t)$ and $\text{Cost}(m^*, t)$ are differentiable because $m(t)$ and $m^*(t)$ are differentiable. We integrate equation (A.41) from time $t = 0$ to $t = T$ to obtain

$$\text{Cost}(m, T) \leq \text{PCR} \cdot \text{Cost}(m^*, T) + \Phi(0) - \Phi(T) \leq \text{PCR} \cdot \text{Cost}(m^*, T) + \Phi(0), \quad (\text{A.42})$$

where the last step follows because $\Phi(T)$ is non-negative. The proof of Proposition 6.1 is therefore completed if we manage to find a differentiable potential function $\Phi(t)$ satisfying equation (A.41) and $\Phi(0) = \frac{\beta \cdot m(0)}{r_2}$.

Choice of $\Phi(t)$. Define the potential function $\Phi(t)$ such that

$$\begin{aligned} \Phi(t) = & \begin{cases} c_5 \beta \cdot (d_{R_1}(t) - m(t) + m^*(t)), & \text{if } m(t) > m^*(t) \\ c_6 \beta \cdot (d_{r_1}(t) - m(t) + m^*(t)) & \text{if } m(t) \leq m^*(t) \end{cases} \\ & + \frac{\beta \cdot m(t)}{r_2} + c_6 R_2 \theta \cdot [q(t) - q^*(t)]^+, \end{aligned} \quad (\text{A.43})$$

where

$$d_r(t) = \sqrt{\frac{r\omega \cdot ([q(t) - q^*(t)]^+)^2}{\beta} + (m(t) - m^*(t))^2}. \quad (\text{A.44})$$

Note that $\Phi(t)$ is non-negative and $\Phi(0) = \frac{\beta \cdot m(0)}{r_2}$. The sophisticated reader might remark that there are points in the domain for which $\Phi(t)$ is not differentiable. As there can only be countably many of these points, these points do not influence the integral of equation (A.41) and we simply ignore these points in the analysis.

Verification of (A.41). We continue by verifying equation (A.41). We distinguish four cases, depending on whether $q(t) > q^*(t)$ or $q(t) \leq q^*(t)$ and $m(t) > m^*(t)$ or $m(t) \leq m^*(t)$.

(i) Assume that $q(t) > q^*(t)$ and $m(t) > m^*(t)$. Recall that, by definition,

$$\frac{dq}{dt} = \lambda(t) - m(t), \quad \frac{dm}{dt} = \frac{\hat{r}_1(t)\omega \cdot q(t) - \hat{r}_2(t)\theta \cdot m(t)}{\beta} \leq \frac{R_1\omega \cdot q(t)}{\beta}. \quad (\text{A.45})$$

The derivative of $d_{R_1}(t)$ is therefore at most

$$\begin{aligned}
\beta \cdot \frac{dd_{R_1}(t)}{dt} &\leq d_{R_1}(t)^{-1} \cdot \left(\begin{aligned} &R_1\omega \cdot (q(t) - q^*(t))(\lambda(t) - m(t)) \\ &+ R_1\omega \cdot (q^*(t) - q(t))(\lambda(t) - m^*(t)) \\ &+ R_1\omega \cdot q(t) \cdot (m(t) - m^*(t)) \\ &+ \beta \cdot \frac{dm^*}{dt} \cdot (m^*(t) - m(t)) \end{aligned} \right) \\
&= d_{R_1}(t)^{-1} \cdot (m(t) - m^*(t)) \left(R_1\omega \cdot q^*(t) - \beta \cdot \frac{dm^*}{dt} \right) \\
&\leq R_1\omega \cdot q^*(t) + \beta \cdot \left[-\frac{dm^*}{dt} \right]^+.
\end{aligned} \tag{A.46}$$

The derivative of the potential function $\Phi(t)$ is then

$$\begin{aligned}
\frac{d\Phi(t)}{dt} &\leq c_5 R_1 \omega \cdot q^*(t) + c_5 \beta \cdot \left(\left[-\frac{dm^*}{dt} \right]^+ + \frac{dm^*}{dt} \right) - \left(c_5 - \frac{1}{r_2} \right) \beta \cdot \frac{dm}{dt} \\
&\quad + c_6 R_2 \theta \cdot (\lambda(t) - m(t) - \lambda(t) + m^*(t)) \\
&\leq c_5 R_1 \omega \cdot q^*(t) + c_5 \beta \cdot \left[\frac{dm^*}{dt} \right]^+ - \left(1 + \frac{1}{r_1} \right) \beta \cdot \frac{dm}{dt} \\
&\quad + \left(1 + R_2 + \frac{R_2}{r_1} \right) \theta \cdot (m^*(t) - m(t)).
\end{aligned} \tag{A.47}$$

The derivative of the cumulative cost $\text{Cost}(m, t)$ is

$$\begin{aligned}
\frac{\partial \text{Cost}(m, t)}{\partial t} &= \omega \cdot q(t) + \beta \cdot \left[\frac{dm}{dt} \right]^+ + \theta \cdot m(t) \\
&\leq \frac{\beta}{r_1} \cdot \frac{dm}{dt} + \beta \cdot \left[\frac{dm}{dt} \right]^+ + \left(1 + \frac{R_2}{r_1} \right) \theta \cdot m(t).
\end{aligned} \tag{A.48}$$

We sum equation (A.47) and (A.48) and cancel terms to obtain

$$\begin{aligned}
\frac{d\Phi(t)}{dt} + \frac{\partial \text{Cost}(m, t)}{\partial t} &\leq c_5 R_1 \omega \cdot q^*(t) + c_5 \beta \cdot \left[\frac{dm^*}{dt} \right]^+ + \left(1 + R_2 + \frac{R_2}{r_1} \right) \theta \cdot m^*(t) \\
&\leq \text{PCR} \cdot \frac{\partial \text{Cost}(m^*, t)}{\partial t}.
\end{aligned} \tag{A.49}$$

Note that if $\frac{dm}{dt} \geq 0$, then the sum follows immediately. If $\frac{dm}{dt} < 0$, we apply the bound

$$-\beta \cdot \frac{dm}{dt} \leq R_2 \theta \cdot m(t) - r_1 \omega \cdot q(t) \leq R_2 \theta \cdot m(t). \tag{A.50}$$

(ii) Assume that $q(t) \leq q^*(t)$ and $m(t) > m^*(t)$. The potential function $\Phi(t)$ simplifies to

$$\Phi(t) = \frac{\beta \cdot m(t)}{r_2}. \tag{A.51}$$

The derivative of the potential function $\Phi(t)$ is then

$$\frac{d\Phi(t)}{dt} = \frac{\beta}{r_2} \cdot \frac{dm}{dt} \leq \frac{R_1 \omega}{r_2} \cdot q^*(t) - \theta \cdot m(t). \tag{A.52}$$

The derivative of the cumulative cost $\text{Cost}(m, t)$ is

$$\begin{aligned}\frac{\partial \text{Cost}(m, t)}{\partial t} &= \omega \cdot q(t) + \beta \cdot \left[\frac{dm}{dt} \right]^+ + \theta \cdot m(t) \\ &\leq \omega \cdot q(t) + \beta \cdot [R_1 \omega \cdot q(t) - r_2 \theta \cdot m(t)]^+ + \theta \cdot m(t) \\ &\leq (1 + R_1) \omega \cdot q^*(t) + \theta \cdot m(t)\end{aligned}\tag{A.53}$$

We sum equation (A.52) and (A.53) and cancel terms to obtain

$$\frac{d\Phi(t)}{dt} + \frac{\partial \text{Cost}(m, t)}{\partial t} \leq \left(1 + R_1 + \frac{R_1}{r_2}\right) \omega \cdot q^*(t) \leq \text{PCR} \cdot \frac{\partial \text{Cost}(m^*, t)}{\partial t}.\tag{A.54}$$

(iii) Assume that $q(t) > q^*(t)$ and $m(t) \leq m^*(t)$. Recall that, by definition,

$$\frac{dq}{dt} = \lambda(t) - m(t), \quad \frac{dm}{dt} = \frac{\hat{r}_1(t) \omega \cdot q(t) - \hat{r}_2(t) \theta \cdot m(t)}{\beta} \geq \frac{r_1 \omega \cdot q(t) - R_2 \theta \cdot m(t)}{\beta}.\tag{A.55}$$

The derivative of $d_{r_1}(t)$ is therefore at most

$$\begin{aligned}\beta \cdot \frac{dd_{r_1}(t)}{dt} &\leq d_{r_1}(t)^{-1} \cdot \begin{pmatrix} r_1 \omega \cdot (q(t) - q^*(t))(\lambda(t) - m(t)) \\ + r_1 \omega \cdot (q^*(t) - q(t))(\lambda(t) - m^*(t)) \\ + (r_1 \omega \cdot q(t) - R_2 \theta \cdot m(t))(m(t) - m^*(t)) \\ + \beta \cdot \frac{dm^*}{dt} \cdot (m^*(t) - m(t)) \end{pmatrix} \\ &\leq d_{r_1}(t)^{-1} \cdot (m^*(t) - m(t)) \left(R_2 \theta \cdot m(t) + \beta \cdot \frac{dm^*}{dt} \right) \\ &\leq R_2 \theta \cdot m(t) + \beta \cdot \left[\frac{dm^*}{dt} \right]^+.\end{aligned}\tag{A.56}$$

The derivative of the potential function $\Phi(t)$ is then

$$\begin{aligned}\frac{d\Phi(t)}{dt} &\leq c_6 R_2 \theta \cdot m(t) + c_6 \beta \cdot \left(\left[\frac{dm^*}{dt} \right]^+ + \frac{dm^*}{dt} \right) - \left(c_6 - \frac{1}{r_2} \right) \beta \cdot \frac{dm}{dt} \\ &\quad + c_6 R_2 \theta \cdot (\lambda(t) - m(t) - \lambda(t) + m^*(t)) \\ &\leq c_6 R_2 \theta \cdot m^*(t) + 2c_6 \beta \cdot \left[\frac{dm^*}{dt} \right]^+ - \left(c_6 - \frac{1}{r_2} \right) \beta \cdot \frac{dm}{dt}.\end{aligned}\tag{A.57}$$

Similar to before, the derivative of the cumulative cost $\text{Cost}(m, t)$ is

$$\frac{\partial \text{Cost}(m, t)}{\partial t} \leq \frac{\beta}{r_1} \cdot \frac{dm}{dt} + \beta \cdot \left[\frac{dm}{dt} \right]^+ + \left(1 + \frac{R_2}{r_1} \right) \theta \cdot m(t).\tag{A.58}$$

We sum equation (A.57) and (A.58) and cancel terms to obtain

$$\begin{aligned}\frac{d\Phi(t)}{dt} + \frac{\partial \text{Cost}(m, t)}{\partial t} &\leq 2c_6 \beta \cdot \left[\frac{dm^*}{dt} \right]^+ + \left(2c_6 R_2 + 1 - \frac{R_2}{r_2} \right) \theta \cdot m^*(t) \\ &\leq \text{PCR} \cdot \frac{\partial \text{Cost}(m^*, t)}{\partial t}.\end{aligned}\tag{A.59}$$

(iv) Assume that $q(t) \leq q^*(t)$ and $m(t) \leq m^*(t)$. The potential function $\Phi(t)$ simplifies to

$$\Phi(t) = 2c_6\beta \cdot (m^*(t) - m(t)) + \frac{\beta \cdot m(t)}{r_2}. \quad (\text{A.60})$$

The derivative of the potential function $\Phi(t)$ is then

$$\frac{d\Phi(t)}{dt} = 2c_6\beta \cdot \frac{dm^*}{dt} - \left(2c_6 - \frac{1}{r_2}\right) \beta \cdot \frac{dm}{dt}. \quad (\text{A.61})$$

Similar to before, the derivative of the cumulative cost $\text{Cost}(m, t)$ is

$$\frac{\partial \text{Cost}(m, t)}{\partial t} \leq \frac{\beta}{r_1} \cdot \frac{dm}{dt} + \beta \cdot \left[\frac{dm}{dt}\right]^+ + \left(1 + \frac{R_2}{r_1}\right) \theta \cdot m(t). \quad (\text{A.62})$$

We sum equation (A.61) and (A.62) and cancel terms to obtain

$$\begin{aligned} \frac{d\Phi(t)}{dt} + \frac{\partial \text{Cost}(m, t)}{\partial t} &\leq 2c_6\beta \cdot \left[\frac{dm^*}{dt}\right]^+ + \left(2c_6R_2 + 1 - \frac{R_2}{r_2}\right) \theta \cdot m^*(t) \\ &\leq \text{PCR} \cdot \frac{\partial \text{Cost}(m^*, t)}{\partial t}. \end{aligned} \quad (\text{A.63})$$

A.8 Proof of Proposition 6.2

Fix a finite time horizon T , arrival rate function $\lambda(\cdot)$ and initial number of servers $m(0)$. Let $\tilde{m}(\cdot)$ be the number of advised servers of AP and $\tilde{q}(\cdot)$ be the corresponding workload. Assume that the advised number of servers achieves finite cost. If the advised number of servers does not achieve finite cost, then Proposition 6.2 follows immediately. Without loss of generality, similar to the proof of Proposition 6.1, assume that $\tilde{m}(t)$ is differentiable.

Overview of the proof. As argued before (see the proof of Proposition 6.1), the proof of Proposition 6.2 requires us to find a non-negative potential function $\Phi(t)$ such that

$$\frac{d\Phi(t)}{dt} + \frac{\partial \text{Cost}(m, t)}{\partial t} \leq \text{PCR} \cdot \frac{\partial \text{Cost}(\tilde{m}, t)}{\partial t}, \quad (\text{A.64})$$

where we have omitted λ from the notation $\text{Cost}^\lambda(m, t)$.

Choice of $\Phi(t)$. Define the potential function $\Phi(t)$ such that

$$\begin{aligned} \Phi(t) &= \begin{cases} c_1\beta \cdot (d_{r_1}(t) - m(t) + \tilde{m}(t)) & \text{if } \hat{r}_1(t) = r_1, \\ c_2\beta \cdot d_{R_1}(t) - c_3\beta \cdot (m(t) - \tilde{m}(t)) & \text{if } \hat{r}_1(t) = R_1, \end{cases} \\ &\quad + \frac{\beta \cdot m(t)}{R_2} + c_4\theta \cdot [q(t) - \tilde{q}(t)]^+, \end{aligned} \quad (\text{A.65})$$

where

$$d_r(t) = \sqrt{\frac{r\omega \cdot ([q(t) - \tilde{q}(t)]^+)^2}{\beta} + (m(t) - \tilde{m}(t))^2}. \quad (\text{A.66})$$

Note that $\Phi(0) = \frac{\beta \cdot m(0)}{R_2}$. If $\hat{r}_1(t) = r_1$ or $m(t) \leq \tilde{m}(t)$ then $\Phi(t)$ is trivially non-negative. Assume that $\hat{r}_1(t) = R_1$ and $m(t) > \tilde{m}(t)$, then

$$\Phi(t) \geq c_2\beta \cdot d_{R_1}(t) - c_3\beta \cdot (m(t) - \tilde{m}(t)) \geq \left(c_2\sqrt{1+2R_1} - c_3\right)\beta \cdot (m(t) - \tilde{m}(t)) \geq 0, \quad (\text{A.67})$$

and hence $\Phi(t)$ is non-negative. The sophisticated reader might remark that there are points in the domain for which $\Phi(t)$ is not differentiable. As there can only be countably many of these points, these points do not influence the integral of equation (A.64) and we simply ignore these points in the analysis.

Verification of (A.64). We continue by verifying equation (A.64). We distinguish eight cases, depending on whether $q(t) > \tilde{q}(t)$ or $q(t) \leq \tilde{q}(t)$, $m(t) > \tilde{m}(t)$ or $m(t) \leq \tilde{m}(t)$ and $\hat{r}_1(t) = r_1$ or $\hat{r}_1(t) = R_1$.

(i.a) Assume that $q(t) > \tilde{q}(t)$, $m(t) > \tilde{m}(t)$ and $\hat{r}_1(t) = r_1$. Note that $\hat{r}_2(t) = r_2$ because $q(t) > \tilde{q}(t)$. Recall that, by definition,

$$\frac{dq(t)}{dt} = \lambda(t) - m(t), \quad \frac{dm(t)}{dt} = \frac{r_1\omega \cdot q(t) - r_2\theta \cdot m(t)}{\beta}. \quad (\text{A.68})$$

The derivative of $d_{r_1}(t)$ is therefore at most

$$\begin{aligned} \beta \cdot \frac{dd_{r_1}(t)}{dt} &\leq d_{r_1}(t)^{-1} \cdot \begin{pmatrix} r_1\omega \cdot (q(t) - \tilde{q}(t))(\lambda(t) - m(t)) \\ + r_1\omega \cdot (\tilde{q}(t) - q(t))(\lambda(t) - \tilde{m}(t)) \\ + (r_1\omega \cdot q(t) - r_2\theta \cdot m(t))(m(t) - \tilde{m}(t)) \\ + \beta \cdot \frac{d\tilde{m}}{dt} \cdot (\tilde{m}(t) - m(t)) \end{pmatrix} \\ &= d_{r_1}(t)^{-1} \cdot (m(t) - \tilde{m}(t)) \left(r_1\omega \cdot \tilde{q}(t) - r_2\theta \cdot m(t) - \beta \cdot \frac{d\tilde{m}}{dt} \right) \\ &\leq r_1\omega \cdot \tilde{q}(t) - \frac{r_2\theta}{\sqrt{1+2r_1}} \cdot m(t) + \beta \cdot \left[-\frac{d\tilde{m}}{dt} \right]^+ - \frac{\beta}{\sqrt{1+2r_1}} \cdot \left[\frac{d\tilde{m}}{dt} \right]^+. \end{aligned} \quad (\text{A.69})$$

The derivative of the potential function $\Phi(t)$ is then

$$\begin{aligned} \frac{d\Phi(t)}{dt} &\leq c_1r_1\omega \cdot \tilde{q}(t) - \frac{c_1r_2\theta}{\sqrt{1+2r_1}} \cdot m(t) \\ &\quad + c_1\beta \cdot \left[-\frac{d\tilde{m}}{dt} \right]^+ - \frac{c_1\beta}{\sqrt{1+2r_1}} \cdot \left[\frac{d\tilde{m}}{dt} \right]^+ + c_1\beta \cdot \frac{d\tilde{m}}{dt} \\ &\quad - \left(c_1 - \frac{1}{R_2} \right) \beta \cdot \frac{dm}{dt} + c_4\theta \cdot (\lambda(t) - m(t) - \lambda(t) + \tilde{m}(t)) \\ &\leq c_1r_1\omega \cdot \tilde{q}(t) - \frac{r_2\theta}{r_1} \cdot m(t) + \left(1 + \frac{1}{R_2} \right) \beta \cdot \left[\frac{d\tilde{m}}{dt} \right]^+ \\ &\quad - \left(1 + \frac{1}{r_1} \right) \beta \cdot \frac{dm}{dt} + c_4\theta \cdot (\tilde{m}(t) - m(t)). \end{aligned} \quad (\text{A.70})$$

The derivative of the cumulative cost $\text{Cost}(m, t)$ is

$$\begin{aligned} \frac{\partial \text{Cost}(m, t)}{\partial t} &= \omega \cdot q(t) + \beta \cdot \left[\frac{dm}{dt} \right]^+ + \theta \cdot m(t) \\ &= \frac{\beta}{r_1} \cdot \frac{dm}{dt} + \beta \cdot \left[\frac{dm}{dt} \right]^+ + \left(1 + \frac{r_2}{r_1} \right) \theta \cdot m(t). \end{aligned} \quad (\text{A.71})$$

We sum equation (A.70) and (A.71) and cancel terms to obtain

$$\begin{aligned} \frac{d\Phi(t)}{dt} + \frac{\partial \text{Cost}(m, t)}{\partial t} &\leq c_1 r_1 \omega \cdot \tilde{q}(t) + \left(1 + \frac{1}{R_2}\right) \beta \cdot \left[\frac{d\tilde{m}}{dt}\right]^+ + c_4 \theta \cdot \tilde{m}(t) \\ &\leq \text{OCR} \cdot \frac{\partial \text{Cost}(\tilde{m}, t)}{\partial t}. \end{aligned} \quad (\text{A.72})$$

Note that if $\frac{dm(t)}{dt} \geq 0$, then the sum follows immediately. If $\frac{dm(t)}{dt} < 0$, we apply the bound

$$-\beta \cdot \frac{dm(t)}{dt} = r_2 \theta \cdot m(t) - r_1 \omega \cdot q(t) \leq r_2 \theta \cdot m(t). \quad (\text{A.73})$$

(i.b) Assume that $q(t) > \tilde{q}(t)$, $m(t) > \tilde{m}(t)$ and $\hat{r}_1(t) = R_1$. Note that $\hat{r}_2(t) = r_2$ because $q(t) > \tilde{q}(t)$. The derivative of $d_{R_1}(t)$ is therefore at most

$$\begin{aligned} \beta \cdot \frac{dd_{R_1}(t)}{dt} &\leq d_{R_1}(t)^{-1} \cdot \begin{pmatrix} R_1 \omega \cdot (q(t) - \tilde{q}(t))(\lambda(t) - m(t)) \\ + R_1 \omega \cdot (\tilde{q}(t) - q(t))(\lambda(t) - \tilde{m}(t)) \\ + R_1 \omega \cdot q(t) \cdot (m(t) - \tilde{m}(t)) \\ + \beta \cdot \frac{d\tilde{m}}{dt} \cdot (\tilde{m}(t) - m(t)) \end{pmatrix} \\ &= d_{R_1}(t)^{-1} \cdot (m(t) - \tilde{m}(t)) \left(R_1 \omega \cdot \tilde{q}(t) - \beta \cdot \frac{d\tilde{m}}{dt} \right) \\ &\leq \frac{R_1 \omega}{\sqrt{1 + 2R_1}} \cdot \tilde{q}(t) + \frac{\beta}{\sqrt{1 + 2R_1}} \cdot \left[-\frac{d\tilde{m}}{dt} \right]^+. \end{aligned} \quad (\text{A.74})$$

The derivative of the potential function $\Phi(t)$ is then

$$\begin{aligned} \frac{d\Phi(t)}{dt} &\leq \frac{c_2 R_1 \omega}{\sqrt{1 + 2R_1}} \cdot \tilde{q}(t) + \frac{c_2 \beta}{\sqrt{1 + 2R_1}} \cdot \left[-\frac{d\tilde{m}}{dt} \right]^+ + c_3 \beta \cdot \frac{d\tilde{m}}{dt} \\ &\quad - \left(c_3 - \frac{1}{R_2} \right) \frac{dm}{dt} + c_4 \theta \cdot (\lambda(t) - m(t) - \lambda(t) + \tilde{m}(t)) \\ &\leq \frac{c_2 R_1 \omega}{\sqrt{1 + 2R_1}} \cdot \tilde{q}(t) + c_3 \beta \cdot \left[\frac{d\tilde{m}}{dt} \right]^+ \\ &\quad - \left(1 + \frac{1}{R_1} \right) \beta \cdot \frac{dm}{dt} + c_4 \theta \cdot (\tilde{m}(t) - m(t)). \end{aligned} \quad (\text{A.75})$$

The derivative of the cumulative cost $\text{Cost}(m, t)$ is

$$\frac{\partial \text{Cost}(m, t)}{\partial t} = \frac{\beta}{R_1} \cdot \frac{dm}{dt} + \beta \cdot \left[\frac{dm}{dt} \right]^+ + \left(1 + \frac{r_2}{R_1} \right) \theta \cdot m(t). \quad (\text{A.76})$$

We sum equation (A.75) and (A.76) and cancel terms to obtain

$$\begin{aligned} \frac{d\Phi(t)}{dt} + \frac{\partial \text{Cost}(m, t)}{\partial t} &\leq \frac{c_2 R_1}{\sqrt{1 + 2R_1}} \omega \cdot \tilde{q}(t) + c_3 \beta \cdot \left[\frac{d\tilde{m}}{dt} \right]^+ + c_4 \theta \cdot \tilde{m}(t) \\ &\leq \text{OCR} \cdot \frac{\partial \text{Cost}(\tilde{m}, t)}{\partial t}. \end{aligned} \quad (\text{A.77})$$

- (ii.a) Assume that $q(t) \leq \tilde{q}(t)$, $m(t) > \tilde{m}(t)$ and $\hat{r}_1(t) = r_1$. Note that $\hat{r}_2(t) = R_2$ because $m(t) > \tilde{m}(t)$ and $q(t) \leq \tilde{q}(t)$. The potential function $\Phi(t)$ simplifies to

$$\Phi(t) = \frac{\beta \cdot m(t)}{R_2} \quad (\text{A.78})$$

The derivative of the potential function $\Phi(t)$ is then

$$\frac{d\Phi(t)}{dt} = \frac{\beta}{R_2} \cdot \frac{dm}{dt} = \frac{r_1 \omega}{R_2} \cdot q(t) - \theta \cdot m(t). \quad (\text{A.79})$$

The derivative of the cumulative cost $\text{Cost}(m, t)$ is

$$\begin{aligned} \frac{\partial \text{Cost}(m, t)}{\partial t} &= \omega \cdot q(t) + \beta \cdot \left[\frac{dm}{dt} \right] + \theta \cdot m(t) \\ &= \omega \cdot q(t) + \beta \cdot \left[\frac{r_1 \omega \cdot q(t)}{\beta} - \frac{R_2 \theta \cdot m(t)}{\beta} \right]^+ + \theta \cdot m(t) \\ &\leq (1 + r_1) \omega \cdot q(t) + \theta \cdot m(t). \end{aligned} \quad (\text{A.80})$$

We sum equation (A.79) and (A.80) and cancel terms to obtain

$$\frac{d\Phi(t)}{dt} + \frac{\partial \text{Cost}(m, t)}{\partial t} \leq \left(1 + r_1 + \frac{r_1}{R_2} \right) \omega \cdot \tilde{q}(t) \leq \text{OCR} \cdot \frac{\partial \text{Cost}(\tilde{m}, t)}{\partial t}. \quad (\text{A.81})$$

- (ii.b) Assume that $q(t) \leq \tilde{q}(t)$, $m(t) > \tilde{m}(t)$ and $\hat{r}_1(t) = R_1$. However, $\hat{r}_1(t) = R_1$ implies that $m(t) - \tilde{m}(t) \leq [q(t) - \tilde{q}(t)]^+ \cdot \sqrt{\frac{\omega}{2\beta}} = 0$ which contradicts our assumption.
- (iii.a) Assume that $q(t) > \tilde{q}(t)$, $m(t) \leq \tilde{m}(t)$ and $\hat{r}_1(t) = r_1$. However, $\hat{r}_1(t) = r_1$ implies that $m(t) - \tilde{m}(t) > [q(t) - \tilde{q}(t)] \cdot \sqrt{\frac{\omega}{2\beta}} \geq 0$ which contradicts our assumption.
- (iii.b) Assume that $q(t) > \tilde{q}(t)$, $m(t) \leq \tilde{m}(t)$ and $\hat{r}_1(t) = R_1$. Note that $\hat{r}_2(t) = r_2$ because $m(t) \leq \tilde{m}(t)$. The derivative of $d_{R_1}(t)$ is therefore at most

$$\begin{aligned} \beta \cdot \frac{dd_{R_1}(t)}{dt} &\leq d_{R_1}(t)^{-1} \cdot \begin{pmatrix} R_1 \omega \cdot (q(t) - \tilde{q}(t))(\lambda(t) - m(t)) \\ + R_1 \omega \cdot (\tilde{q}(t) - q(t))(\lambda(t) - \tilde{m}(t)) \\ + (R_1 \omega \cdot q(t) - r_2 \theta \cdot m(t))(m(t) - \tilde{m}(t)) \\ + \beta \cdot \frac{d\tilde{m}}{dt} \cdot (\tilde{m}(t) - m(t)) \end{pmatrix} \\ &\leq d_{R_1}(t)^{-1} \cdot (\tilde{m}(t) - m(t)) \left(r_2 \theta \cdot m(t) + \beta \cdot \frac{d\tilde{m}}{dt} \right) \\ &\leq r_2 \theta \cdot m(t) + \beta \cdot \left[\frac{d\tilde{m}}{dt} \right]^+. \end{aligned} \quad (\text{A.82})$$

The derivative of the potential function $\Phi(t)$ is then

$$\begin{aligned}
\frac{d\Phi(t)}{dt} &\leq c_2 r_2 \theta \cdot m(t) + c_2 \beta \cdot \left[\frac{d\tilde{m}}{dt} \right]^+ + c_3 \beta \cdot \frac{d\tilde{m}}{dt} \\
&\quad - \left(c_3 - \frac{1}{R_2} \right) \beta \cdot \frac{dm}{dt} + c_4 \theta \cdot (\lambda(t) - m(t) - \lambda(t) + \tilde{m}(t)) \\
&\leq c_2 r_2 \theta \cdot m(t) + (c_2 + c_3) \beta \cdot \left[\frac{d\tilde{m}}{dt} \right]^+ \\
&\quad - \left(1 + \frac{1}{R_1} \right) \beta \cdot \frac{dm}{dt} + c_4 \theta \cdot (\tilde{m}(t) - m(t))
\end{aligned} \tag{A.83}$$

The derivative of the cumulative cost $\text{Cost}(m, t)$ is

$$\frac{\partial \text{Cost}(m, t)}{\partial t} = \frac{\beta}{R_1} \cdot \frac{dm}{dt} + \beta \cdot \left[\frac{dm}{dt} \right]^+ + \left(1 + \frac{r_2}{R_1} \right) \theta \cdot m(t). \tag{A.84}$$

We sum equation (A.83) and (A.84) and cancel terms to obtain

$$\begin{aligned}
\frac{d\Phi(t)}{dt} + \frac{\partial \text{Cost}(m, t)}{\partial t} &\leq (c_2 + c_3) \beta \cdot \left[\frac{d\tilde{m}}{dt} \right]^+ + c_4 \theta \cdot \tilde{m}(t) \\
&\leq \text{OCR} \cdot \frac{\partial \text{Cost}(\tilde{m}, t)}{\partial t}.
\end{aligned} \tag{A.85}$$

(iv.a) Assume that $q(t) \leq \tilde{q}(t)$, $m(t) \leq \tilde{m}(t)$ and $\hat{r}_1(t) = r_1$. However, $\hat{r}_1(t) = r_1$ implies that $m(t) - \tilde{m}(t) > [q(t) - \tilde{q}(t)]^+ \cdot \sqrt{\frac{\omega}{2\beta}} = 0$ which contradicts our assumption.

(iv.b) Assume that $q(t) \leq \tilde{q}(t)$, $m(t) \leq \tilde{m}(t)$ and $\hat{r}_1(t) = R_1$. Note that $\hat{r}_2(t) = r_2$ because $m(t) \leq \tilde{m}(t)$. The potential function $\Phi(t)$ simplifies to

$$\Phi(t) = (c_2 + c_3) \beta \cdot (\tilde{m}(t) - m(t)) + \frac{\beta \cdot m(t)}{R_2} \tag{A.86}$$

The derivative of the potential function $\Phi(t)$ is then

$$\frac{d\Phi(t)}{dt} = (c_2 + c_3) \beta \cdot \frac{d\tilde{m}}{dt} - \left(c_2 + 1 + \frac{1}{R_1} \right) \beta \cdot \frac{dm}{dt}. \tag{A.87}$$

The derivative of the cumulative cost $\text{Cost}(m, t)$ is

$$\frac{\partial \text{Cost}(m, t)}{\partial t} = \frac{\beta}{R_1} \cdot \frac{dm}{dt} + \beta \cdot \left[\frac{dm}{dt} \right]^+ + \left(1 + \frac{r_2}{R_1} \right) \theta \cdot m(t). \tag{A.88}$$

We sum equation (A.87) and (A.88) and cancel terms to obtain

$$\begin{aligned}
\frac{d\Phi(t)}{dt} + \frac{\partial \text{Cost}(m, t)}{\partial t} &\leq (c_2 + c_3) \beta \cdot \left[\frac{d\tilde{m}}{dt} \right]^+ + c_4 \theta \cdot \tilde{m}(t) \\
&\leq \text{OCR} \cdot \frac{\partial \text{Cost}(\tilde{m}, t)}{\partial t}.
\end{aligned} \tag{A.89}$$

A.9 Proof of Proposition 4.12

Fix any algorithm \mathcal{A} , and let $\text{CR}(\eta)$ denote its competitive ratio when it has access to an η -accurate prediction. Fix $\delta > 0$ and assume that $\text{CR}(0) \leq 1 + \delta$. We will construct an instance for which $\text{Cost}^\lambda(m, T) \geq \text{OPT}/(4\delta^2) + \Phi(0)$. Throughout the example, we will assume that $\Phi(0)$ is zero without loss of generality.

Let $T = 2 + \sqrt{2}\delta$, $m(0) = 2$ and $\lambda(t) = 4$ for $t \in [0, \sqrt{2}\delta]$. The value of $\lambda(t)$ for $t \in [\sqrt{2}\delta, T]$ will be specified later. Fix $\beta = \omega = 1$ and $\theta = 0$. Let the prediction $\tilde{\lambda}(t) = 4$ for $t \in [0, T]$ and let $m(t)$ be the number of servers of \mathcal{A} for the instance. We distinguish two cases depending on the value of $m(\sqrt{2}\delta)$.

1. First, consider the case when $m(\sqrt{2}\delta) < 3$. Fix $\lambda(t) = 4$ for $t \in [\sqrt{2}\delta, T]$. The optimal solution of (2.1) is $m^*(t) = 4$ for $t \in [0, T]$. The value of the optimal solution is purely due to switching cost and is equal to $\text{OPT} = 2$. As $\tilde{\lambda}(t) = \lambda(t)$, the prediction $\tilde{\lambda}$ is perfect. Hence, by the assumption that \mathcal{A} is $(1 + \delta)$ -consistent, we must have $\text{Cost}^\lambda(m, T) \leq (1 + \delta)\text{OPT}$ for this instance. We will verify this.

At time $t = \sqrt{2}\delta$, the queue length of \mathcal{A} is at least $q(\sqrt{2}\delta) \geq \int_0^{\sqrt{2}\delta} \lambda(t) - m(t) dt > \sqrt{2}\delta$. The optimal solution starting from time $t = \sqrt{2}\delta$ is $m(t) = 4 + q(\sqrt{2}\delta)/\sqrt{2} > 4 + \delta$ for $t \in (\sqrt{2}\delta, T]$. As a result, the flow-time is at least $\int_0^T q(t) dt \geq q(\sqrt{2}\delta)/\sqrt{2} > \delta$. The cost of \mathcal{A} is therefore at least $\text{Cost}^\lambda(m, T) > 2 + 2\delta \geq (1 + \delta)\text{OPT}$. This is a contradiction with our assumption and the next case must occur.

2. Next, consider the case when $m(\sqrt{2}\delta) \geq 3$. Fix $\lambda(t) = 0$ for $t \in [\sqrt{2}\delta, T]$. The optimal solution of (2.1) is $m^*(t) = 2$ for $t \in [0, T]$ if δ is sufficiently small. The queue length satisfies $q^*(t) = 2t$ for $t \in [0, \sqrt{2}\delta]$ and $q^*(t) = 2\sqrt{2}\delta - 2(t - \sqrt{2}\delta)$ for $t \in [\sqrt{2}\delta, T]$. The value of the optimal solution is purely due to flow-time and is equal to $\text{OPT} = 2\delta^2 + 2\delta^2 = 4\delta^2$.

The cost of \mathcal{A} is at least $\text{Cost}^\lambda(m, T) \geq 1 \geq \text{OPT}/(4\delta^2)$ due to switching cost. Moreover, the mean absolute error (MAE) is $T \cdot \|\tilde{\lambda} - \lambda\|_{\text{MAE}} = \int_0^T |\tilde{\lambda}(t) - \lambda(t)| dt = 8$ and hence the prediction is $2/\delta^2$ -accurate.

Hence, equation (4.16) follows.

A.10 Proof of Proposition 4.13

Let $m(t)$ denote the number of servers of the Timer algorithm. We will construct an instance $(T, \lambda, m(0))$ for which $\text{Cost}^\lambda(m, T) \geq \text{OPT}/(2\omega) + \Phi(0)$ for any $\Phi(0)$.

Let $T = n\omega^{-1}$, $m(0) = \omega^{-1}$ and $\lambda(t) = \sum_{i=0}^{n-1} \omega^{-1} \cdot \delta_{i\omega^{-1}}(t)$, where $\delta_{t_0}(t)$ is the Dirac delta at t_0 , i.e. $\delta_{t_0}(t_0) = \infty$, $\delta_{t_0}(t) = 0$ for all $t \neq t_0$ and $\int_0^\infty \delta_{t_0}(s) ds = 1$. Fix $\beta = \omega^{-1}$ and $\theta = 1$. Let $m(t)$ be the number of servers of the Timer algorithm for the instance. Since the time between subsequent arrivals is $\omega^{-1} = \beta/\theta$, the Timer Algorithm does not turn off any servers. Hence, the number of servers is $m(t) = \omega^{-1}$ for $t \in [0, T]$. The cost is therefore

$$\text{Cost}^\lambda(m, T) = \theta T \cdot \omega^{-1} = n\omega^{-2}. \quad (\text{A.90})$$

Let $m^*(t) = 1$ for $t \in [0, T]$. The corresponding queue length satisfies $q^*(i\omega^{-1} + t) = \omega^{-1} - t$ for $t \in [0, \omega^{-1})$ and $i = 0, \dots, n-1$. The cost of the optimal solution is at most the cost of this

algorithm, i.e.

$$\begin{aligned}\text{OPT} &\leq \omega \cdot \int_0^T q^*(t) \, dt + \theta \cdot \int_0^T m^*(t) \, dt = n\omega \cdot \int_0^{\omega^{-1}} (\omega^{-1} - t) \, dt + \theta \cdot \int_0^T 1 \, dt \\ &= n\omega \cdot \frac{\omega^{-2}}{2} + \theta T = \frac{3n\omega^{-1}}{2}.\end{aligned}\tag{A.91}$$

Let $n \geq 4\omega^2 \cdot \Phi(0)$ and therefore

$$\text{COST}^\lambda(m, T) = n\omega^{-2} \geq \frac{3n\omega^{-2}}{4} + \Phi(0) \geq \frac{1}{2\omega} \cdot \text{OPT} + \Phi(0),\tag{A.92}$$

which proves the statement.