Online Scheduling with Multi-State Machines*

Dawsen Hwang‡ Patrick Jaillet§

August 2015

Abstract

In this paper, we propose a general framework capturing online scheduling problems in which each machine has multiple states that lead to different processing times. For these problems, in addition to decide how to assign jobs to machines, we also need to set the states of the machines each time they are assigned jobs. For a wide range of machine environments, job processing characteristics and constraints, and cost functions, we develop a 5.14-competitive deterministic online algorithm and a 3.65-competitive randomized online algorithm.

The online weighted traveling repairman problem belongs to this general framework, and both our deterministic and randomized online algorithms lead to lower competitive ratios than the current existing ones in the literature. In addition, we show that the online algorithm ReOpt (re-optimizing the route of the repairman whenever a new request is released) is almost surely asymptotically optimal for a probabilistic version of this problem.

1 Introduction

In an online scheduling problem, jobs are released over time and an *online algorithm*, knowing only the jobs released so far, is to assign jobs to machines in an online fashion so as to minimize a given cost function of the completion time of the jobs. *Competitive analysis* is a standard performance metric for evaluating an online algorithm, and is based on the concept of *competitive ratio*, which, in our context, is defined as the worst-case ratio (among all problem instances) between the overall cost of the online algorithm and that of the optimal offline algorithm which has the full knowledge ahead of time about all jobs to be released. A lower competitive ratio implies a better online algorithm.

Unlike offline scheduling problems, one limitation in the majority of the online scheduling literature is the assumption that the processing time of a job depends only on its assigned machine. This assumption is not realistic in applications where each machine has multiple states that lead to different processing times. This is the case for many manufacturing problems such as producing paper bags, semiconductors, and automobiles [2, 33, 47, 51]. For example, Pinedo [47] describes an application in a paper bag factory, where a machine has different states, each corresponding to the size of bags

*Research funded in part by AFOSR grant FA9550-10-1-0437 and ONR grant N00014-12-1-0033

‡Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, 02139; dawsen@mit.edu

§Department of Electrical Engineering and Computer Science, and Operations Research Center, MIT, Cambridge, MA, 02139; jaillet@mit.edu

and the combination of colors the machine can produce. Problems involving multiple machine states are typically studied in the offline (but not the online) setting and under the simplification that each job can only be processed in the minimal-processing-time state. However, this simplification fails to consider the state-transition time, which may also be important in an overall time-objective. In this paper, we study the more general set-up where we remove the above-mentioned simplification, and where we consider an online setting. In particular, as we will see, our algorithms will need to balance the tradeoff between minimizing the processing time and the state-transition time.

1.1 Our Contributions

Our contributions consist in the following three main aspects.

First, we formulate a new class of online scheduling problems. In this setting, each machine has a state that can be controlled over time and the processing time of jobs depends on the machine states. In addition, we introduce a family of generic cost functions for these problems that can describe many practical objectives such as minimizing the makespan and minimizing the total weighted completion time. The offline version of this new online scheduling problem is NP-hard, because it includes the Weighted Traveling Repairman Problem (WTRP) as a special case. Therefore, solving the offline version efficiently is challenging *per se*. In this paper, we restrict our attention to quantifying the competitive ratio and do not consider the computational complexity aspect of an online algorithm.

Second, for this new generic class of scheduling problems, we derive deterministic and randomized online algorithms using a plan-and-commit approach, where algorithms plan the schedule of jobs and the control of machine states at predefined geometric time steps, and commit to the plan (regardless of newly released jobs between these time steps). The analyses of the specific plan-and-commit-based online algorithms in the classical online scheduling and online WTRP literature use a summation-transformation proof technique. However, this proof technique cannot be applied to our versions of algorithms, which makes it difficult to obtain a provable low competitive ratio. In this paper, we tackle this challenge by using a factor-revealing-linear-program-based proof technique, and obtain a low-competitive-ratio online algorithm for all problems in the general framework. Our competitive ratios are better than the existing ones for the online WTRP (a special case in the general framework) in both deterministic and randomized settings.

Third, for a probabilistic version of the online WTRP, we prove that an existing algorithm ReOpt (re-optimizing the route of the server whenever a new request is revealed in a greedy fashion) is almost surely asymptotically optimal as the number of requests approaches infinity. This is the first theoretical guarantee of the performance of ReOpt for the online WTRP in any performance metrics.

1.2 Related Work

There has been extensive research on problems related to ours. Here we give a review of relevant work, organized around three main categories, as follows.

Scheduling Problems

The area of scheduling has a rich literature. The central question in a scheduling problem is to determine at what time and to which machine each job is assigned so

as to minimize a given cost function. Different machine environments (single machine, parallel (identical) machines, or unrelated machines, etc.), processing characteristics and constraints (preemption/preemption-repeat/non-preemption, release date constraints, precedence constraints, etc.), and cost functions (makespan, total completion time, etc.) result in different versions of scheduling problems. The reader is referred to [22] and [47] for comprehensive reviews.

The predominant models in the scheduling literature assume the processing time of a job to be a function of the machine-job pair assignment. There are two models that do not require this assumption, and they are mostly studied in the offline case. The first such model corresponds to scheduling problems with controllable processing time (for surveys, see [45, 48]). In this model, the processing time of a job depends on the resource allocated to the machine, and costs for allocating resources are imposed. The second such model corresponds to scheduling problems with sequence-dependent set-up time (for surveys, see [1, 2]). In this model, the overall processing time of a job depends on the job processed by the machine right before it. Kim and Bobrowski [34] and Vinod and Sridharan [50] study this last model in a dynamic setting, when jobs arrive over time and obtain simulation-based results. However, to the best of our knowledge, no previous work along these two directions has considered a competitive analysis for online algorithms when the processing time of a job is not a function of the machine-job pair, except for the special cases of online Vehicle Routing Problems VRPs (discussed later). We study general online scheduling problems whose offline setting goes beyond the sequence-dependent processing time model, and design online algorithms with provable competitive ratios.

Online scheduling has been studied extensively for the basic model where the processing time of a job is a function of the machine-job pair. Among these problems, most relevant to our work are the ones with the cost function being the total (weighted) completion time of all jobs [3, 15, 16, 20, 21, 24, 25, 38, 42, 43, 46, 49]. The goal in this line of research is to derive online algorithms with the smallest competitive ratios. Recently, Günther *et al.* [23] develop an algorithmic approach that can approximate the best-possible competitive ratios for many versions of these problems. We note here that the solution techniques described in all the papers discussed in this paragraph cannot, in general, be adapted to fit our problems because the effect of the machine states on the processing time is not taken into account. The only exceptions are [14, 24, 25] whose ideas can be used to design algorithms for our problems (elaborated more in the solution techniques part of this section). As we will see later, our algorithm involves other design ideas and is not a trivial generalization of those contained in [14, 24, 25].

Online Vehicle Routing Problems

In the online VRPs, a server with a unit speed limit is to visit requests that are located in a metric space and released over time. The server is originally located at a prescribed depot. Different characteristics of requests (weights, precedence constraints, capacity constraints, etc.), cost functions (the makespan, total weighted completion time, etc.), and underlying metric spaces (the nonnegative real line, the real line, general continuous metric spaces, discrete metric spaces, etc) result in different problems. The reader is referred to [29] for a survey.

The online VRPs are special cases of our new class of scheduling problems when we view the servers as the machines and the location of a server as the state of the corresponding machine. The online VRPs are more restrictive because when a machine

(a server) finishes processing a job (a request), the machine state (the server location) always corresponds to the location of that job and thus cannot be selected by the algorithm. The specific class of cost functions we study here covers classical online VRPs such as the online WTRP (discussed more later), the online nomadic traveling salesman problem [4, 8, 11, 37], and the nomadic version of the online quota traveling salesman problem [6, 7, 26, 27], the nomadic and/or the latency online dial-a-ride problems (with infinite capacity) [13, 17, 37] (see Remark 1 for detail). Therefore, our general online algorithms can be applied to the above online VRPs. Our algorithms and competitive analyses can also be applied to the nomadic and/or the latency online dial-a-ride problems with capacity constraints. Moreover, both our deterministic and randomized online algorithms achieve better competitive ratios than the existing ones for the online WTRP and the latency online dial-a-ride problem (with or without capacity constraints).

The online WTRP is a version of the online VRP in which the objective is to minimize the total weighted completion time. We provide here a more detailed literature review about the online WTRP because the technical aspect of this problem is relevant to ours. Feuerstein and Stougie [17] propose a deterministic 9-competitive online algorithm and show that no deterministic online algorithm has a competitive ratio lower than 2.41 when the metric space is the non-negative real line. With some minor modifications, the 9-competitive online algorithm becomes 3.5-competitive when the metric space is the non-negative real line [37]. Feuerstein and Stougie [17] also prove that 2 and 2.33 are lower bounds on the competitive ratios of any randomized online algorithm for the real line and for general metric spaces respectively. Jaillet and Wagner [28], Krumke *et al.* [36] propose two different deterministic online algorithms with a competitive ratio of 5.83 for general metric spaces. The randomized version of these two online algorithms both have a competitive ratio of 3.87.

The latency online dial-a-ride problem is a variant of the WTRP. In this problem, a request contains a source location and a destination location, and the server needs to transport the request from the source to the destination. The 5.83-competitive deterministic and 3.87-competitive randomized online algorithms for the online WTRP can also be applied to the latency online dial-a-ride problems (with or without capacity constraints) [28, 36]. For the online latency online dial-a-ride problem with capacity one, Feuerstein and Stougie [17] prove that any deterministic online algorithm has a competitive ratio of at least 3 when the metric space contains the real line.

An interesting algorithm for the online WTRP is ReOpt, which re-optimizes and follows the route that minimizes the total weighted completion time of unserved requests whenever a new request is released. ReOpt belongs to the class of *zealous algorithms*, defined by Blom *et al.* [11], in which the server always travels with the maximum speed when there are unserved requests and may change directions only when either it arrives at a request location or a new request is released. To the best of our knowledge, the only zealous algorithm for which a competitive analysis has been applied is a 6.04-competitive deterministic online algorithm proposed by Ausiello *et al.* [9] for the real line. Therefore, it is unknown whether the competitive ratio of ReOpt is finite or not, even for the real line. Jaillet and Wagner [30] attempt to give the only performance guarantee for ReOpt. In particular, they try to show that, under some stochastic assumptions, ReOpt is *almost surely asymptotically optimal*, i.e., it has a competitive ratio of 1 almost surely when the number of requests goes to infinity. However, one part of their proof (Lemma 5 in [30]) turns out to be flawed, invalidated their overall argument. We provide here a full proof

that ReOpt is indeed almost surely asymptotically optimal.

Solution Techniques

The core idea in the design of our algorithms is to divide time into geometric steps, and control the machine states and assign jobs to machines at each time step via solving a specific auxiliary (offline) problem. Jaillet and Wagner [28], Krumke *et al.* [36] apply this idea with an auxiliary problem consisting of maximizing the total weight of the requests to be completed by the next time step, and derive a 5.83-competitive deterministic algorithm and a 3.87-competitive randomized algorithm for the online WTRP. When applied to the classical online scheduling problems $1|r_j|\sum w_j c_j$ and $P|r_j|\sum w_j c_j$, the idea of using such an auxiliary problem leads to a 4-competitive deterministic algorithm and a 2.89-competitive randomized algorithm [14, 24, 25]. The competitive ratios of these algorithms are obtained based on a summation-transformation proof approach, which compares the cost of the online algorithm and that of the optimal offline algorithm using the following transformation of summation: $\sum w_j c_j = \int_{x=0}^{\infty} \sum_{j:c_j \geq x} w_j \, \mathrm{d}x$. This transformation provides an expression suitable for comparing the costs between the online and the optimal offline algorithms because at each time step, the total weight of requests not completed by a deadline (proportional to the time step itself) of the solution to the auxiliary offline problem is at most that of the optimal offline algorithm. This particular auxiliary problem and the summation-transformation proof technique have also been applied to derive polynomial-time approximated algorithms for the (offline) TRP [12, 19].

Instead of minimizing the actual completion time, the above auxiliary problem constraints it with a deadline, which is a major drawback due to its "low resolution". Ideally, in addition to maximizing the total weight that can be completed by the next time step, we want the auxiliary offline problem to simultaneously minimize the total weighted completion time of requests that are completed by the next time step. For the classical online scheduling problem $1|r_j|\sum w_j C_j$, Hall *et al.* [25] address this issue by using a two-stage optimization approach: finding the optimal order for serving those jobs so as to minimize the total weighted completion time after solving the original auxiliary problem. The modified schedule is feasible for $1|r_j|\sum w_j C_j$ because the total processing time does not depend on the order in which the jobs are scheduled. With this two-stage optimization approach, Hall *et al.* [25] obtain a 3-competitive deterministic algorithm for $1|r_j|\sum w_j C_j$. However, this approach does not have an analogy for the online WTRP because the completion time of the last request depends on the order in which the requests are served and thus the order that minimizes the total weighted completion time can be infeasible with respect to the deadline constraint. Here we attempt to solve two optimization problems simultaneously by setting the objective functions to be the summation of the two original ones. At a conceptual level, Koutsoupias and Papadimitriou [35] use this idea in the design of the famous $(2k-1)$-competitive Work Function Algorithm for the online k-server problem, but their approach bears no similarity to ours on a technical level.

The summation-transformation proof technique described earlier does not provide useful competitive ratios for our algorithms. Therefore, we derive a competitive analysis using *factor-revealing Linear Programs (LPs)*, i.e., LPs whose objective values correspond to the quantity of interest (in our case, an upper bound on the competitive ratio). Although the factor-revealing-LP approach has been used to calculate the approximation ratio or competitive ratio of algorithms in many problems [5, 18, 31, 32, 39–41, 44],

its application to our problems remains non-trivial because of the particular choice of suitable variables, objectives, and constraints for factor-revealing LPs strategies to become successful.

1.3 Organization

The rest of the paper is organized as follows.

In Section 2, we formulate a new class of online scheduling problems where each machine has multiple states, and the processing time of jobs depends on the states of the machines. In addition, we define a new family of cost functions including many classical ones such as the makespan and the total weighted completion time. Finally, we show that the online VRPs are special cases of the newly defined online scheduling problems.

In Section 3, we consider the online WTRP because the solution to this special case provides intuition on how to approach the general case. We define a novel family of parameterized online algorithms, analyze the algorithms, and find the parameters that give the lowest provable competitive ratios. By doing so, we obtain a 5.14-competitive deterministic online algorithm and a 3.65-competitive randomized online algorithm. The analysis is not tight and the lower bounds on the competitive ratios of the above two algorithms are 4 and 2.82 respectively. Finally, we consider a probabilistic formulation of the online WTRP and prove that ReOpt is almost surely asymptotically optimal as the number of requests approaches infinity.

In Section 4, we consider problems in the general framework, i.e., online scheduling with multi-state machines when the cost belongs to our new family of cost functions (following some minor technical assumptions, as discussed in Appendix A). The analysis for the parameterized online algorithms designed for the online WTRP, in general, cannot be applied to the general setting, because the cost functions are not necessarily linear in the completion times of jobs. Therefore, we construct online algorithms for the general problems based on the parameters that lead to the best provable competitive ratios for the online WTRP, and analyze only the resulting online algorithms. By doing so, we obtain 5.14-competitive deterministic and 3.65-competitive randomized online algorithms.

Finally, in Section 5, we summarize our results and conclude with four open problems.

2 Problem Formulation

In Section 2.1, we formally describe the *online scheduling problem with multi-state machines*, a new class of online scheduling problems. The defining feature of this new problem is that each machine has multiple states and the processing time of jobs depends on the state of the machine processing it. In addition, we introduce a generic class of cost functions, which we call *the total costs of active projects*. These generic cost functions cover many classical ones such as the makespan and the total weighted completion time. In Section 2.2, we formally describe the online WTRP. Moreover, we show that the online WTRP is a special case of the new class of scheduling problems.

2.1 Online Scheduling with Multi-State Machines

In Section 2.1.1, we formulate the basic setting of the *online scheduling problem with multi-state machines*. In Section 2.1.2, we describe several settings, i.e., machine environments and job processing characteristics and constraints, to which our online algorithms and analysis can be applied.

2.1.1 Basic Setting

Machines

We assume that we have m machines, indexed by $i \in [m] \triangleq \{1, 2, \ldots, m\}$, where each machine i can process at most one job at a time. Each machine i has an internal state s_i that can be controlled over time $t \in \mathbb{R}_{\geq 0}$. The state of machine i is assumed to take values in a metric space $(\mathbb{M}_i, d_i)$, with the initial state being a prescribed origin $O_i \in \mathbb{M}_i$. The distance $d_i(s_i^1, s_i^2)$, $s_i^1, s_i^2 \in \mathbb{M}_i$, is defined to be the minimum time required for the state of machine i to change from s_i^1 to s_i^2. When being directed to go from state s_i^1 to s_i^2, machine i commits itself to a time duration of $d_i(s_i^1, s_i^2)$ and cannot process any job during that period.

Problem Instances and Jobs

A problem instance I is composed of a finite set of n jobs, indexed by $j \in [n]$ where the size n is part of input. Each job j is characterized by a release date $r_j \in \mathbb{R}_{\geq 0}$, a function $p_{ij} : \mathbb{M}_i \to \mathbb{R}_{\geq 0}$ of processing time for each machine $i \in [m]$, and some other characteristics ρ_j belonging to a set P (to be defined later):

- The release date r_j is the earliest time at which any machine can start processing job j. Without loss of generality, we assume $0 \leq r_1 \leq r_2 \leq \cdots \leq r_n$.
- The function p_{ij}, $i \in [m]$, is defined such that for any state $s_i \in \mathbb{M}_i$, $p_{ij}(s_i)$ is the required time for machine i to process job j when in state s_i. In the basic setting, we do not allow preemption, meaning that once machine i in state s_i starts processing job j at time t, $t \in \mathbb{R}_{\geq 0}$, both the machine i and the job j commit themselves to an amount of time duration $p_{ij}(s_i)$: the machine i cannot process any other job and the job j cannot be processed by any other machine during that period. Also, we assume we cannot control the machine state when it is processing any jobs, and processing a job does not change the machine state. As a result, the completion time of job j will be $c_j \triangleq t + p_{ij}(s_i)$.
- The other characteristics ρ_j are used to define the cost, which we will discuss next.

Cost Functions

Here we introduce a new generic class of cost functions, which we call the *total costs of active projects*. Before describing the mathematical formulation, we first introduce the intuition behind the new cost functions and then illustrate the idea with two examples.

In our setting, the overall cost is the summation of the costs contributed by *projects*. The list of projects is given in advance irrespective of the problem instance. Each project corresponds to completing a set of jobs whose characteristics collectively satisfy some conditions and is said to be *active* when the released jobs so far are included in the set defining that project. Only active projects are counted in he overall cost. The cost of an active project is a function of the completion time of that project (the earliest time

when all jobs in the set defining the project are completed). For simplicity, we call the above completion-time-to-cost function the *cost function* of that project. A project's cost function is realized when it becomes active, using the characteristics of jobs released so far. Note that the characteristics of the jobs determine both the "activeness" and the cost function of each project. Therefore, we need to define P and $\rho_j \in P$, $j \in I$, accordingly (recall that $\rho_j \in P$ is the set of characteristics of job j other than r_j and $\{p_{ij}\}_{i=1}^m$).

This new class of cost functions covers many classical ones, including the following two examples:

1. The total weighted completion time ($\sum w_j c_j$).
2. The quota-collecting makespan, i.e., the earliest time when the total value ($v_j \in \mathbb{R}_{>0}$) of completed jobs achieves a prescribed quota $Q \in \mathbb{R}_{>0}$. This corresponds to $\min_{S|\sum_{j\in S} v_j \geq Q} \max_{j\in S} c_j$.

In the first example, a (possibly countably infinite) number of projects $\mathbb{N}_{\geq 1}$ are involved, where project $j \in \mathbb{N}_{\geq 1}$ corresponds to completing a particular job j. Project j is active if and only if job j is in the problem instance. Following the definition, the completion time of an active project j is the same as the completion time of the job j. At the time when project j becomes active, w_j is revealed to online algorithms, and the cost function is defined to be $h_j(x) = w_j x, x \in \mathbb{R}_{\geq 0}$. Because we need w_j to calculate the cost function of project j, w_j is included as one of the characteristic of job j, setting $P = \mathbb{R}_{>0}$ and $\rho_j = w_j$.

In the second example, there is only one project, which consists of completing a set of jobs whose total value achieves/exceeds a given quota Q. The project is active if there exists a subset of jobs in the instance whose total value achieves/exceeds the quota. We need the value v_j of each job j to determine whether the project is active. Therefore, we include v_j in the characteristics of a job by setting $P = \mathbb{R}_{>0}$ and $\rho_j = v_j$. If the project is active, then the cost function is simply the identity function.

Now let us formally describe the definition behind the *total costs of active projects*. In this setting, a (finite or infinite) collection of projects K is given a priori, independent of the problem instance. Each project $k \in K$ is defined by a *satisfying set indicator* $\mathbb{1}_k$, which defines whether a particular set of jobs' characteristics collectively satisfy the conditions specified by project k; and a *cost function* h_k, which relates the completion time of project k to the amount it contributes to the overall cost:

- The satisfying set indicator $\mathbb{1}_k$ of a project $k \in K$ is a binary function that maps a subset S of jobs with corresponding characteristics $(\rho_j | j \in S)$ to whether the characteristics of jobs in S collectively satisfy the conditions specified by project k or not. Note that in a problem instance, there might be zero, one, or multiple sets S such that $\mathbb{1}_k(S, (\rho_j | j \in S)) = 1$. For each problem instance I, we denote $K(I)$ as the set of all active projects, or mathematically,

$$K(I) \triangleq \{k | k \in K, \exists S \subset I \text{ such that } \mathbb{1}_k(S, (\rho_j | j \in S)) = 1\}.$$

  The completion time of an active project $k \in K(I)$, denoted x_k, is defined to be the earliest time at which all jobs in one of the satisfying set are completed, or

mathematically,

$$x_k \triangleq \min_{S\subset I, \mathbb{1}_k(S,(\rho_j|j\in S))=1} \max_{j\in S} c_j.$$

- The cost function h_k of a project $k \in K(I)$ relates the completion time x_k to the cost contributed by project k. As discussed earlier, the cost function h_k can depend on the following characteristics of jobs:

$$(\rho_j|j \in I, r_j \leq \min_{S\subset I, \mathbb{1}_k(S,(\rho_j|j\in S))=1} \max_{j'\in S} r_{j'})$$

  where the min max notation is the time when project k becomes active (recall that r_j is the release date of job j).

The cost defined by the total costs of active projects is then

$$\text{cost}(I) \triangleq \sum_{k\in K(I)} h_k(x_k) = \sum_{k\in K(I)} h_k(\min_{S\subset I, \mathbb{1}_k(S,(\rho_j|j\in S))=1} \max_{j\in S} c_j).$$

In Table 1, we describe how the following four classical cost functions can be described using our framework: the total weighted completion time ($\sum_j w_j c_j$), the quota-collecting makespan ($\min_{S|\sum_{j\in S} v_j \geq Q} \max_{j\in S} c_j$), the discounted total weighted completion time ($\sum_j w_j(1-e^{-rc_j})$), and the makespan ($\max_j c_j$).

Table 1: Four classical cost functions formulated as the total costs of active projects.

Cost	K	P	ρ_j	$S\|\mathbb{1}_k(S,(\rho_j\|j\in S))=1$	$h_k(x)$
$\sum_j w_j c_j$	$\mathbb{N}_{\geq 1}$	$\mathbb{R}_{>0}$	w_j	$S=\{k\}$	$\rho_k x$
$\min_{S\|\sum_{j\in S} v_j\geq Q} \max_{j\in S} c_j$	$\{1\}$	$\mathbb{R}_{>0}$	v_j	$\sum_{j\in S}\rho_j \geq Q$	x
$\sum_j w_j(1-e^{-rc_j})$	$\mathbb{N}_{\geq 1}$	$\mathbb{R}_{>0}$	w_j	$S=\{k\}$	$\rho_k(1-e^{-rx})$
$\max_j c_j$	$\mathbb{N}_{\geq 1}$	$\emptyset$		$S=\{1,2,\ldots,k\}$	$r^k x$ $(r\to\infty)$

The total costs of active projects can also describe other cost functions of practical interest, such as in the following example. Assume a given company owning the machines makes an amount $v_j \in \mathbb{R}_{>0}$, $j \in [m]$, of money when completing job j. For each $k \in \mathbb{N}$, when earning a total amount $Q_k \in \mathbb{R}_{>0}$ of money, the company offers a corresponding bonus $w_j \in \mathbb{R}_{>0}$ to its employee. Because the employee operating the machines wishes to get the bonus as early as possible, his/her goal is to minimize the weighted summation of the times at which the total money earned achieves each level of Q_k, $k \in \mathbb{N}$, for those Q_ks that are achievable. In this example, the projects can be indexed by $K = \mathbb{N}_{\geq 1}$ and project k, $k \in K$, corresponds to making a total amount of Q_k of money. Regarding the characteristics of jobs, we set $P = \mathbb{R}_{>0}$ and $\rho_j = v_j$. A set of jobs S satisfies the conditions set by project k, i.e., $\mathbb{1}_k(S,(\rho_j|j \in S)) = 1$, if and only if $\sum_{j\in S}\rho_j \geq Q_k$. Finally, a project $k \in K(I)$ contributes an amount of $w_k x_k$ to the cost, and hence the cost function is defined to be $h_k(x) = w_k x$.

In this paper, for our results to hold, we need to impose the following assumption on the cost functions:

Assumption 1. *For any $k \in K(I)$, h_k is non-decreasing and concave, and $h_k(0) = 0$.*

We assume that h_k is non-decreasing because a greater completion time for a project does not decrease its cost for most practical applications. We assume $h_k(0) = 0$ because a project completed at the start should not contribute any cost. Concavity captures many practical cost functions, including the four in Table 1 and the additional one we described earlier. This assumption is applicable to problems where the unit-time cost for a project k until completion is decreasing in time, that is, when $h_k(t+\delta) - h_k(t)$ is decreasing in t for any $\delta > 0$.

Algorithms

In this paper, an algorithm determines the states of the machines and the schedule of jobs over time $t \in \mathbb{R}_{\geq 0}$ subject to the constraints regarding machines and jobs described earlier. Based on the available information of the problem instances at time t, we classify the algorithms into *offline* algorithms that know the entire problem instance I from the start, and *online* algorithms that know only jobs with release dates up to time t, i.e., $I_t \triangleq \{(r_j, \{p_{ij}\}_{i=1}^m, \rho_j) | r_j \leq t\}$. Our goal is to design online algorithms with strong worst-case (over all problem instances) performance in comparison with the optimal offline algorithm (formalized in Definition 2.1.1). Thus, we are only concerned with an online algorithm which is well-defined mathematically, and not about its computational complexity.

Online algorithms can be further classified as *deterministic* and *randomized* online algorithms. A deterministic online algorithm determines the machines states and the assignment of jobs at time t as a function of I_t. A randomized online algorithm is a distribution over a collection of deterministic online algorithms.

For notational convenience, for a deterministic (online or offline) algorithm with name alg, we denote c_j^{alg} the corresponding completion time of job $j \in I$ and x_k^{alg} the corresponding completion time of project $k \in K(I)$. In addition, we denote the cost of problem instance I under algorithm alg as

$$\text{alg}(I) \triangleq \sum_{k \in K(I)} h_k\left(x_k^{\text{alg}}\right).$$

For a randomized online algorithm with name rand who is a distribution over the collection of deterministic algorithms $\{\text{alg}(\omega) : \omega \in \Omega\}$, the cost of problem instance I, denoted $\text{rand}(I)$, is defined to be the expected cost, i.e., $\text{rand}(I) \triangleq \mathbb{E}_{\omega \sim \Omega}(\text{alg}(\omega)(I))$.

For each problem instance I, the optimal offline algorithm OPT is the one that achieves the infimum of the costs among all algorithms, i.e.,

$$\text{OPT}(I) \triangleq \inf_{\text{alg}} \text{alg}(I).$$

Here we use the infimum rather than the minimum in the expression of $\text{OPT}(I)$ because we use its value as a baseline for evaluating the online algorithms, but are not concerned about whether this value can be achieved by any specific offline algorithm and how this value can be computed. However, we can show that the infimum is achievable under some technical assumptions (see Appendix A) using an argument similar to that of Lemma 16.

In this paper, we wish to design online algorithms whose costs are "close" to $\text{OPT}(I)$. This performance metric can be formalized as the *competitive ratio*, as described below:

Definition (Competitive Analysis)**.** *An online algorithm OnA is c-competitive,* $c \in \mathbb{R}_{\geq 1}$*, if for any problem instance I,*

$$OnA(I) \leq cOPT(I).$$

The competitive ratio *of OnA is the infimum of c such that OnA is c-competitive. Smaller competitive ratios imply better online algorithms.*

2.1.2 Problem Examples

In addition to the basic setting described above, our results can be applied to many general settings of online scheduling with multi-state machines. In this section, we describe several such problem examples using the conventional three-fold notation $\alpha|\beta|\gamma$ [22, 47]. The α field represents the machine environment, the β field represents the processing characteristics and constraints, and the γ field represents the cost function. Since we are interested in only the case where the cost function is the total costs of active projects, we introduce only the machine environments (α field) and processing characteristics and constraints (β field) to which our results can be applied. As a side note, the greek letters α, β, γ refer to problem examples in this section only but they do not carry the same meaning in the other parts of this paper.

Machine Environments

In the basic setting described in Section 2.1.1, the machine environment corresponds to unrelated machines in parallel, meaning that each job needs to be processed by only one machine, and the processing times on different machines do not necessary have any relations.

The machine environments that our algorithms and analysis can be applied to includes *flow shop*, in which each job has to be processed by all machines in a prescribed and job-independent order; *job shop*, in which each job has to be processed by all machines in a job-specific order; and *open shop*, in which each job has to be processed by all machines in any order.

Processing Characteristics and Constraints

In the basic setting described in Section 2.1.1, there are two processing constraints. The first is the release date constraint, meaning that the earliest time a job can start being processed is its release date. This constraint is necessary for our results to be applicable. The second is the non-preemption constraint, meaning that we cannot interrupt the processing of a job. This constraint can be replaced with the *preemption-repeat* constraint, under which the processing of a job can be interrupted but has to start form scratch if resumed later.

In addition to the above constraints, we can add *precedence constraints* to the β field, allowing each job to specify a set of jobs that has to be completed before the job itself can start being processed.

Last but not least, we can add *batch processing (batch*(b)*,* $b \in \mathbb{N}_{\geq 2}$*)* as a processing characteristic to the β field, allowing a machine to process a batch of up to b jobs at the same time. The processing time of a batch of jobs is the maximum processing time among the jobs in that batch, and the completion times of all jobs in the same batch are defined as the completion time of that batch. If preemption-repeat is allowed and the processing of a batch is interrupted, all jobs in that batch are not completed and have to be processed from scratch if being processed later.

2.2 Online Weighted Traveling Repairman Problem (WTRP).

In this section, we show that the online WTRP is a special case of the general framework described in Section 2.1.

To begin with, we first describe the online WTRP. In this problem, a single server (the repairman), initially located at a depot $D \in \mathbb{M}$, travels in a metric space $(\mathbb{M}, d)$ with a unit speed limit in order to visit requests located in the metric space. A problem instance consists of a finite list of n requests, indexed as $1, 2, \ldots, n$, where n is instance-specific. Each request $j \in [n]$ has a release date $r'_j \in \mathbb{R}_{\geq 0}$, a location $l_j \in \mathbb{M}$, and a weight $w'_j \in \mathbb{R}_{>0}$. The completion time of a request is the earliest time at which the server arrives at the location of the request after or at its release date. The objective is to minimize the total weighted completion time.

Here is how we can formulate the online WTRP as a problem in our general framework. In this formulation, there is only one machine, i.e., $m = 1$. The machine state represents the server location, i.e., $(\mathbb{M}_1, d_1) = (\mathbb{M}, d)$ and $O_1 = D$. Each request corresponds to a job with the same release date. The location of a request is included in the definition of the processing time of the corresponding job: when the state of the machine is at the location of the request, the processing time is 0; otherwise, the processing time is infinity, i.e.,

$$p_{1j}(x) = \begin{cases} 0 & \text{if } x = l_j, \\ \infty & \text{if } x \neq l_j. \end{cases}$$

The characteristics of jobs and the projects are defined so that the cost is the total weighted completion time, as described in Table 1.

Remark 1. *The release date and the processing time in the above formulation is applicable to all variants of the VRPs. If we restrict our attention to those with cost functions being in Table 1, we can obtain the following two variants: First, when the cost functions is the makespan, the problem becomes the online nomadic traveling salesman problem. Second, when the cost functions is the quota-collecting makespan, the problem becomes the nomadic version of the online quota traveling salesman problem.*

If we allow precedence constraints in the problem formulation, our general framework can cover the nomadic and/or the latency online dial-a-ride problems (with infinite capacity). In any of these two problems, a request j is characterized by (r'_j, u_j, v_j, w'_j), where r'_j and w'_j are the release date and the weights, and u_j and v_j are the source and destination locations of the request, and the request has to be delivered from u_j to v_j. Our general framework can describe this request by setting two jobs $2j-1$ and and $2j$, where $r_{2j-1} = r_{2j} = r'_j$,

$$p_{1,2j-1}(x) = \begin{cases} 0 & \textit{if } x = u_j, \\ \infty & \textit{if } x \neq u_j, \end{cases}$$

$$p_{1,2j}(x) = \begin{cases} 0 & \textit{if } x = v_j, \\ \infty & \textit{if } x \neq v_j, \end{cases}$$

$w_{2j-1} = 0$ and $w_{2j} = w'_j$, and impose the precedence constraint that we cannot start job $2j$ until we have completed job $2j-1$. We can describe the nomadic and the latency online dial-a-ride problems by setting the cost functions to be the makespan and the total weighted completion time respectively.

2.2.1 Probabilistic Version

Here we describe a probabilistic version of the online WTRP, where we make the following stochastic assumptions:

Assumption 2 (Locations)**.** *The metric space is an M-dimensional Euclidean space, $M \in \mathbb{N}_{\geq 1}$, and the locations are independently drawn from an identical distribution over a compact support in the metric space.*

Assumption 3 (Release Dates)**.** *For all $j \in [n]$, $r_j = \sum_{i=1}^{j} Y_i$ where $Y_1, Y_2, \ldots, Y_n$ are independent random variables drawn from an identical distribution of a non-negative support with a finite mean and variance.*

Assumption 4 (Weights)**.** *The weight of each request has positive upper and lower bounds, i.e., there exists $0 < \omega < \Omega$ such that for all $j \in [n]$, $w_j \in [\omega, \Omega]$.*

3 The Online Weighted Traveling Repairman Problem

In this section, we study the online WTRP, formulated in Section 2.2. In Section 3.1, we propose a family of deterministic and randomized online algorithms, parameterized by a pair of variables (α, β) where $\alpha \in (0, 1]$ and $\beta \in [\alpha, \infty)$. In Section 3.2, we provide upper and lower bounds on the competitive ratios of the proposed algorithms, and determine the parameters (α, β) that lead to the smallest provable competitive ratios. As a result of our analysis, we derive a 5.14-competitive deterministic algorithm and a 3.65-competitive randomized algorithm, both of their competitive ratios are lower than the best existing ones in the literature. When the cost functions of projects are linear, the results above can be modified and applied to the general case described in Section 2.1. However, for general cost functions of projects that satisfy Assumption 1, the design and analysis of online algorithms are both more involved so the results presented in this section cannot be applied directly (see Section 4 for further discussions). Finally, in Section 3.3, we study a probabilistic version of the online WTRP, and show that ReOpt is almost surely asymptotically optimal.

3.1 Deterministic and Randomized Online Algorithms

In this section, we propose a family of deterministic online algorithms (α, β)-Plan-and-Commit ($\text{PAC}_{\alpha,\beta}$) and randomized versions $\text{RPAC}_{\alpha,\beta}$ for each pair of real numbers (α, β) such that $\alpha \in (0, 1]$ and $\beta \in [\alpha, \infty)$.

Both $\text{PAC}_{\alpha,\beta}$ and $\text{RPAC}_{\alpha,\beta}$ have two major stages: initialization and iterations. The randomized algorithm $\text{RPAC}_{\alpha,\beta}$ takes a randomized step only at the end of the initialization stage, but otherwise is the same as $\text{PAC}_{\alpha,\beta}$. Thus, in the following, we focus on describing $\text{PAC}_{\alpha,\beta}$, but we will point out the randomized step taken by $\text{RPAC}_{\alpha,\beta}$.

Initialization

This stage begins at time 0 end ends at a time t_1, which is defined/computed by $\text{PAC}_{\alpha,\beta}$ (or $\text{RPAC}_{\alpha,\beta}$). The server stays at the depot D through the entire period of this stage. Thus, what the online algorithm does in this stage is to compute t_1, which must be greater than zero but cannot be too large. More precisely, for reasons that will be clear when we analyze the algorithms, the value t_1 must satisfy the following two conditions:

1. The time t_1 needs to be non-zero to ensure the geometric series $\{t_l \triangleq t_1(1+2\alpha)^{l-1}\}_{l=1}^{\infty}$ to be unbounded (recall that we have assumed $\alpha > 0$ so $1 + 2\alpha > 1$).

2. No request j can be completed before time $t_1/(1+2\alpha)$ by any algorithm (including an optimal offline algorithm) except for the trivial case where $r_j = 0$ and $l_j = D$.

The online algorithms calculate t_1 as follows. Let τ be a time variable updated by $\text{PAC}_{\alpha,\beta}$ that will end up being t_1 at the end of this stage. At time $t = 0$, $\text{PAC}_{\alpha,\beta}$ sets τ to be either the distance between the depot and the nearest (but not at the depot) request with release date 0, or ∞ if there are no requests with release date 0. Note that because the server has a unit speed limit, we can set the value of the time variable τ to be a distance.

After time 0, if no request is released before τ, then $\text{PAC}_{\alpha,\beta}$ sets $t_1 = \tau$. Otherwise, $\text{PAC}_{\alpha,\beta}$ sets t_1 to be the earliest release date of such requests. For the randomized algorithm $\text{RPAC}_{\alpha,\beta}$, after getting the geometric time series $\{t_l\}_{l=1}^{\infty}$ similarly to what the deterministic algorithm $\text{PAC}_{\alpha,\beta}$ does, the algorithm multiplies each term by the same random variable $(1 + 2\alpha)^{\omega}$ for all l, i.e., setting $t_l \leftarrow t_l \times (1 + 2\alpha)^{\omega}$, where ω is drawn from $[0, 1)$ uniformly once for all l. Clearly, Condition 1 is satisfied according to the way we define t_1. We prove that Condition 2 is also satisfied later in Lemma 3.

Before starting the first iteration of the second stage at time t_1, we define R_1 to be the set of all requests with release date at most t_1, i.e., $R_1 \triangleq I_{t_1}$.

Iterations

For any positive integer l, the l^{th} iteration begins at time t_l and ends at time t_{l+1}. The algorithm is called Plan-and-Commit because it plans the route at time t_l and is committed to following the route until time t_{l+1}, regardless of the requests released between time t_l and t_{l+1}.

At time t_l, the online algorithm calculates what an offline algorithm alg_l would have done between time 0 and αt_l such that the following cost is minimized:

$$\sum_{j \in R_l} w_j f_{\alpha,\beta}(c_j, t_l)$$

where

$$f_{\alpha,\beta}(x, y) \triangleq \begin{cases} x & \text{if } x \le \alpha y \\ \beta y & \text{if } x > \alpha y \end{cases}$$

and the set R_l represents the remaining requests at time t_l and is defined either in the initialization stage (if $l = 1$) or in the previous iteration (if $l \ge 2$). We call the problem of finding the algorithm alg_l the auxiliary offline problem (see Remark 2 for discussions regarding why we choose this specific auxiliary offline problem). It is clear that such an algorithm alg_l exists because the number of permutation of jobs in R_l is finite. If alg_l is computed approximately, then the analysis in Section 3.2 cannot be applied due to Lemma 4 except for cases where $\beta = \infty$ (see Remark 7 for further discussions). For what follows, alg_l is an optimal solution to the auxiliary offline problem.

The auxiliary offline problem can be interpreted as the following. We first view αt_l as the deadline of the algorithm. Then, we view the cost as the summation of contributions of requests in R_l. More precisely, request j in R_l contributes one of the following two amounts to the cost:

- the weighted completion time $w_j c_j$, if completed by the deadline αt_l; or
- a weighted penalty $w_j \beta t_l$ (where βt_l is the penalty), if not completed by the deadline αt_l.

Thus, the auxiliary problem has the objective of both

- minimizing the total weighted completion time; and
- maximizing the total weight

of requests that are completed by the next time step.

The server follows a delayed version of the route of alg_l (delayed by t_l) for a duration of αt_l starting at time t_l. The server then travels through the reverse of the route of alg_l and returns to the depot D at time $(1+2\alpha)t_l = t_{l+1}$. We denote A_l those requests in R_l that have a completion time no greater than αt_l under the offline algorithm alg_l (see Remark 3 for comments regarding the sets $\{A_l\}_{l=1}^{\infty}$). Following the definition, any j in A_l, has a completion time $c_j^{\text{PAC}_{\alpha,\beta}}$ such that

$$c_j^{\text{PAC}_{\alpha,\beta}} \leq t_l + c_j^{\text{alg}_l} \tag{1}$$

At time t_{l+1}, we finish the l^{th} iteration by defining R_{l+1} to be R_l minus A_l plus the requests with release dates in the time interval $(t_l, t_{l+1}]$, i.e., $R_{l+1} \triangleq (R_l \backslash A_l) \cup (I_{t_{l+1}} \backslash I_{t_l})$.

Remark 2. *At each iteration l, we take into account the release dates when finding the auxiliary offline algorithm alg_l even though all requests in A_l have been released by the time we compute the route and can be visited at any time after t_l. The intuition is the following. The release date is a lower bound of the completion time of the request in any offline algorithm, so the optimal offline algorithm would (conceptually) serve requests with lower release dates earlier. By taking into account the release date, alg_l and $PAC_{\alpha,\beta}$ would (conceptually) serve those requests earlier, and thus a lower competitive ratio may be achieved. In fact, taking the release dates into account is necessary for Lemma 7, which is essential to our analysis.*

Remark 3. *The sets $\{A_l\}_{l=1}^{\infty}$ form a partition of all requests I (Lemma 1). The primary goal of the auxiliary offline is to find the partition $\{A_l\}$, which is crucial in the analysis of the competitive ratio. In fact, at each iteration l, if we replace algorithm alg_l by an alternative algorithm that minimizes the total completion time of requests in A_l subject to the constraints that all requests in A_l are completed and the server returns at the depot D at time t_{l+1}, then the upper bound on the competitive ratio still holds.*

3.2 Competitive Analysis

The primary result we have for $\text{PAC}_{\alpha,\beta}$ is the following:

Theorem 1. *The competitive ratio of $PAC_{1,1}$ is between 4 and 5.14.*

The primary result we have for $\text{RPAC}_{\alpha,\beta}$ is the following (see Remark 4 for discussions regarding why our randomized online algorithms have lower competitive ratios than the best existing ones):

Theorem 2. *For $\beta \geq 5$, the competitive ratio of $RPAC_{1,\beta}$ is between $1 + \frac{2}{\ln(3)} \approx 2.82$ and $\frac{4}{\ln(3)} \approx 3.64$.*

The above two major results follows from the following more general theorems. We start by introducing theorems related to the upper bounds on the competitive ratios of $PAC_{\alpha,\beta}$.

For fixed α, when β is relatively large, our best provable upper bound on the competitive ratios has the following closed-form expression:

Theorem 3. *For $\beta \geq 2\alpha^2 + 3\alpha$, the competitive ratio of $PAC_{\alpha,\beta}$ is at most $\frac{(1+2\alpha)(1+\alpha)}{\alpha}$.*

For general β, the lowest provable upper bound is the objective value of a factor-revealing LP, described in the theorem below:

Theorem 4. *For any positive integer N, the objective value of $LP^{det}_{\alpha,\beta}(N)$ is an upper bound on the competitive ratio of $PAC_{\alpha,\beta}$ where $LP^{det}_{\alpha,\beta}(N)$ is defined as follows:*

$$\begin{aligned}
&max\ C_N + T_0 && && (2)\\
&s.t.\ C_{i+1} - C_i \geq \frac{i\alpha}{N}(T_i - T_{i+1}) && \text{for } i = 0, \cdots, N-1 && (3)\\
&\quad C_{i+1} - C_i \leq \frac{(i+1)\alpha}{N}(T_i - T_{i+1}) && \text{for } i = 0, \cdots, N-1 && (4)\\
&\quad 1 \geq \frac{2\alpha^2}{(1+2\alpha)(\beta-\alpha)}C_N + \frac{\alpha(\beta-\alpha(1+2\alpha))}{(1+2\alpha)(\beta-\alpha)}T_0 && (\text{if } \beta > \alpha) && (5)\\
&\quad 1 \geq \frac{\alpha}{2\beta}C_N + \frac{\alpha}{2+4\alpha}T_0. && && (6)\\
&\quad C_N + \frac{\beta - 2\frac{i\alpha}{N}}{1+2\alpha}T_0 \leq C_i + \beta T_i + C_{\lceil \frac{N-2i}{1+2\alpha} \rceil} + \frac{\beta - 2\frac{i\alpha}{N}}{1+2\alpha}T_{\lfloor \frac{N-2i}{1+2\alpha} \rfloor} && \text{for } i = 0, \cdots, \left\lfloor \frac{N}{2} \right\rfloor && (7)\\
&\quad C_i \geq 0 && \text{for } i = 1, \cdots, N\\
&\quad T_i \geq 0 && \text{for } i = 0, \cdots, N-1\\
&\quad C_0 = T_N = 0.
\end{aligned}$$

Theorem 4 allows us to select parameters α and β such that the upper bound on the competitive ratio is minimized. We calculate the objective value of $\mathrm{LP}^{det}_{\alpha,\beta}(N)$ for all pairs of α, β when they are both multiples of 0.01 using Gurubi. For example, Figure 1 illustrates the objective value of $\mathrm{LP}^{det}_{1,\beta}(N)$ when β varies.

Our observation of the numerical results indicates that the lowest upper bound on the competitive ratio among different α and β occurs when $(\alpha, \beta) = (1, 1)$. As a result, we conjecture that $(\alpha, \beta) = (1, 1)$ is the parameter that minimizes the upper bound provided by Theorem 4. The objective value of the LP when $N = 10000$ is slightly below 5.135. Taken into account possible numerical errors, we conclude that the upper bound part of Theorem 1 holds. Moreover, our observation of the numerical results

indicates that when $\beta \geq 2\alpha^2 + 3\alpha$, $\mathrm{LP}^{det}_{\alpha,\beta}(N) \to (1+\alpha)(1+2\alpha)/\alpha$ as $N \to \infty$, which implies that Theorem 4 does not give a better upper bound than Theorem 3.

When it comes to lower bounds, we have the following result for $\mathrm{PAC}_{\alpha,\beta}$:

Theorem 5. *For the case where the metric space $\mathbb{M}$ contains the real line, for any $\alpha \in (0,1]$ and $\beta \in [\alpha, \infty)$, the competitive ratio of $PAC_{\alpha,\beta}$ is at least $3 + \frac{1}{\alpha}$; and as $\beta \to \infty$, the competitive ratio of $PAC_{\alpha,\beta}$ is at least $\frac{(1+\alpha)(1+2\alpha)}{\alpha}$.*

The second statement of Theorem 5 together with Theorem 3 show that the analysis is tight when $\beta \to \infty$.

Similarly, we have the following results for the randomized online algorithms $\mathrm{RPAC}_{\alpha,\beta}$:

Theorem 6. *When $\beta \geq 2\alpha^2 + 3\alpha$, the competitive ratio of $RPAC_{\alpha,\beta}$ is at most $\frac{2(1+\alpha)}{\ln(1+2\alpha)}$.*

Theorem 7. *For any positive integer N, the objective value of $LP^{rand}_{\alpha,\beta}(N)$ is an upper bound on the competitive ratio of $RPAC_{\alpha,\beta}$ where $LP^{rand}_{\alpha,\beta}(N)$ is defined by $LP^{det}_{\alpha,\beta}(N)$ with constraint (5) replaced by the following constraint:*

$$1 \geq \frac{\alpha \ln(1+2\alpha)}{\beta - \alpha} C_N + \frac{(\beta - \alpha(1+2\alpha)) \ln(1+2\alpha)}{2(\beta - \alpha)} T_0 \ (\textit{if } \beta > \alpha). \tag{8}$$

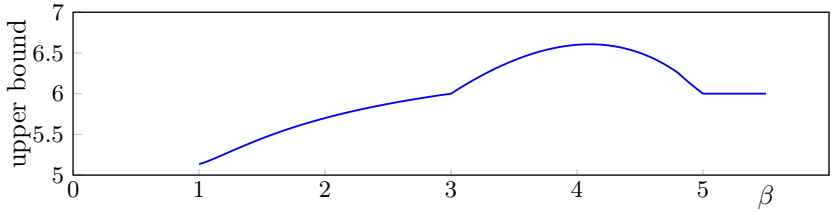
Similar to the case of the deterministic algorithms, we find the objective value of $\mathrm{LP}^{rand}_{\alpha,\beta}(N)$ for all pairs of α, β when they are both multiples of 0.01. For example, Figure 2 displays the objective value of the linear program for $\mathrm{LP}^{rand}_{1,\beta}(N)$ when β varies.

Our observation of the numerical results indicates that the lowest upper bound on the competitive ratio among different α and β occurs at all $(1, \beta)$ with $\beta \geq 5$. Therefore, we conjecture that for all $\beta \geq 5$, $(1, \beta)$ minimizes the provable competitive ratio in our analysis. According to Theorem 3, for $\beta \geq 5$, the competitive ratio of $\mathrm{RPAC}_{1,\beta}$ is at most $4/\ln(3) \approx 3.64$. Hence we have obtained the upper bound part of Theorem 2. The lower bound of Theorem 2 is a special case of the following theorem:

Theorem 8. *For the case where the metric space $\mathbb{M}$ contains the real line, for any $\alpha \in (0,1]$ and $\beta \in [\alpha, \infty)$, the competitive ratio of $RPAC_{\alpha,\beta}$ is at least $1 + \frac{2\alpha}{\ln(1+2\alpha)}$; and as $\beta \to \infty$, the competitive ratio of $RPAC_{\alpha,\beta}$ is at least $\frac{2(1+\alpha)}{\ln(1+2\alpha)}$.*

The rest of this section is organized as follows: In Section 3.2.1, we prove the results related to the upper bounds on the competitive ratio, i.e., Theorems 3, 4, 6, and 7. In Section 3.2.2, we provide adversarial problem instances that prove Theorems 5 and 8.

Figure 1: Upper bounds on the competitive ratio of $\mathrm{PAC}_{1,\beta}$



Remark 4. *When $\beta \geq 2\alpha^2 + 3\alpha$, the expressions of the upper bounds on the competitive ratios for our proposed algorithms are the same as that of the existing competitive online algorithms INTERVAL [36] and BREAK [28], in both deterministic and randomized cases, if one chooses the same common ratio for the geometric time steps. However, we achieve lower competitive ratios, because our algorithms have a wider range ($(1, 3]$) of possible common ratios to choose from, compared to the existing algorithms do ($\left(1, 1+\sqrt{2}\right]$). The existing algorithms solve an auxiliary offline problem right after the server completes a delayed version of the solution given by the auxiliary offline algorithm in the previous time step. However, because the online algorithm must be compared to the optimal offline algorithm that starts from the depot, in the analysis, the auxiliary offline solution is compared with one that returns to the depot through the shortest path and then follows the optimal solution (of the auxiliary offline problem). By doing so, the online algorithm cannot use the information revealed before returning to the depot, which ultimately limits the range of possible common ratios between time steps. On the other hand, our algorithm addresses this issue by solving the auxiliary offline problem after the server returns to the depot, which leads to a wider range of possible common ratios between time steps.*

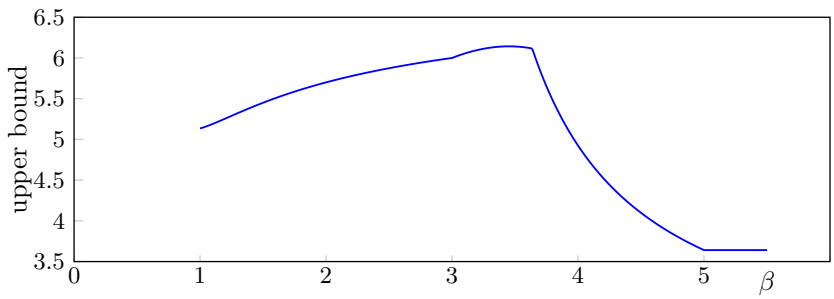
3.2.1 Upper Bounds on The Competitive Ratios

We first claim that, for the sake of determining the upper bound on the competitive ratio, we can assume that no request j has both $r_j = 0$ and $l_j = D$. The reason is the following. If there is a request j such that $r_j = 0$ and $l_j = D$, then for any feasible algorithm, $c_j = 0$. Therefore, removing this request does not change the cost of any (online or offline) algorithm. Hence, without loss of generality, we assume no request j has both $r_j = 0$ and $l_j = D$.

Let us begin by describing the outline of proofs of Theorems 3, 4, 6 and 7. First, we show that $\{A_l\}_{l=1}^{\infty}$ forms a partition of all requests I (Lemma 1). Because of Lemma 1 and (1), for any problem instance problem instance I, $\mathrm{PAC}_{\alpha,\beta}(I)$ has the following upper bound:

$$\mathrm{PAC}_{\alpha,\beta}(I) \leq \sum_{l=1}^{\infty} \sum_{j \in A_l} w_j c_j^{\mathrm{alg}_l} + \sum_{l=1}^{\infty} \sum_{j \in A_l} w_j t_l = C(\alpha)(I) + T(0)(I)$$

Figure 2: Upper bounds on the competitive ratio of $\mathrm{RPAC}_{1,\beta}$



where for all $r \in [0, \alpha]$,

$$C(r)(I) \triangleq \sum_{l=1}^{\infty} \sum_{j \in A_l, c_j^{\mathrm{alg}_l} \le r t_l} w_j c_j^{\mathrm{alg}_l} \qquad \text{and} \qquad T(r)(I) \triangleq \sum_{l=1}^{\infty} \sum_{j \in A_l, c_j^{\mathrm{alg}_l} > r t_l} w_j t_l$$

(see Remark 5 for discussions regarding the definitions). For the case of the randomized algorithm $\mathrm{RPAC}_{\alpha,\beta}(I)$, we use the same notation $C(r)(I)$ and $T(r)(I)$ to represent the expected value of the corresponding functions. As a result, for both $\mathrm{PAC}_{\alpha,\beta}$ and $\mathrm{RPAC}_{\alpha,\beta}$, $\sup_I \frac{C(\alpha)(I)}{\mathrm{OPT}(I)} + \frac{T(0)(I)}{\mathrm{OPT}(I)}$ is an upper bound on the competitive ratio. We drop the parameter I and view $\{\frac{C(r)}{\mathrm{OPT}}\}_{r\in[0,\alpha]}$ and $\{\frac{T(r)}{\mathrm{OPT}}\}_{r\in[0,\alpha]}$ as variables in two factor-revealing LPs, one for $\mathrm{PAC}_{\alpha,\beta}$ and the other for $\mathrm{RPAC}_{\alpha,\beta}$. We then find linear inequalities between $T(r)$, $C(r)$ and OPT that are valid for all problem instances I, and translates those inequalities into linear constraints in the LP. Unless explicitly mentioned, the inequalities we find are valid for any $\omega \in [0, 1)$, and thus are valid for both $\mathrm{PAC}_{\alpha,\beta}$ and $\mathrm{RPAC}_{\alpha,\beta}$. In some cases, the inequalities are valid for only the expected values of $C(r)$ and $T(r)$ when ω is drawn uniformly from $[0, 1)$, and thus those inequalities are valid for only the randomized online algorithms $\mathrm{RPAC}_{\alpha,\beta}$. There are uncountably infinite many variables $\{\frac{C(r)}{\mathrm{OPT}}\}_{r\in[0,\alpha]}$ and $\{\frac{T(r)}{\mathrm{OPT}}\}_{r\in[0,\alpha]}$, so we cannot solve the original LP numerically. Therefore, after obtaining the linear inequalities, we obtain $\mathrm{LP}^{det}_{\alpha,\beta}(N)$ and $\mathrm{LP}^{rand}_{\alpha,\beta}(N)$ by dividing $[0, \alpha]$ into $N + 1$ arithmetic steps , and for all $i = 0, 1, \ldots, N$, we use C_i to represent $C(\frac{i\alpha}{N})/\mathrm{OPT}$ and T_i to represent $T(\frac{i\alpha}{N})/\mathrm{OPT}$. By definition, $C_0 = T_N = 0$, so there is only a total of $2N$ nonnegative real variables $\{C_i\}_{i=1}^{N}$ and $\{T_i\}_{i=0}^{N-1}$.

Let us briefly describe how we obtain the linear inequalities that are used as constraints in the LPs. The first sets of linear inequalities are a direct result of the definition of $\{T(r)\}_{r\in[0,\alpha]}$ and $\{C(r)\}_{r\in[0,\alpha]}$ (Lemma 2) and is independent of the online algorithm, which implies constraints (3) and (4). We then prove that for both $\mathrm{PAC}_{\alpha,\beta}$ and $\mathrm{RPAC}_{\alpha,\beta}$, t_1 is small enough and satisfies Condition 2 in Section 3.1 (Lemma 3), which is essential for the other linear inequalities. The main idea of the rest of the linear inequalities is the observation that for all l, the offline algorithm alg_l is also optimal if the summation of the auxiliary (offline) problem was taken over any set of requests satisfying some properties (Lemma 4). Comparing the cost of the auxiliary problem between alg_l and that of the optimal offline algorithm OPT, we obtain two sets of linear inequalities between $C(\alpha)$, $T(0)$, and OPT (Lemmas 5 and 6), which imply constraints (5) (for $\mathrm{PAC}_{\alpha,\beta}$) or (8) (for $\mathrm{RPAC}_{\alpha,\beta}$) and (6) respectively. Theorems 3 and 6 can be derived from Lemmas 2 and 5. Finally, comparing the cost of the auxiliary problem between alg_l and algorithms that are combinations of alg_l and alg_{l+1} (Figure 3), we obtain linear inequalities between $\{C(r)\}_{r\in[0,\alpha]}$ and $\{T(r)\}_{r\in[0,\alpha]}$ (Lemma 7), which imply constraints (7).

Let us prove the lemmas in detail. To begin, we show that $\{A_l\}_{l=1}^{\infty}$ forms a partition of I.

Lemma 1. *The sequence of subsets $\{A_l\}_{l=1}^{\infty}$ is a partition of I, i.e., $\bigcup_{l=1}^{\infty} A_l = I$ and for any $i \neq j$, $A_i \cap A_j = \emptyset$.*

Proof. It is clear from the definition that for all $i \neq j$, $A_i \cap A_j = \emptyset$, so it is sufficient to show that $I = \bigcup_{l=1}^{\infty} A_l$.

The statement $I = \bigcup_{l=1}^{\infty} A_l$ is equivalent to saying that there exists an integer $\bar{l}$ such that there are no requests to serve after the $\bar{l}^{\text{th}}$ iteration, i.e., $R_l = \emptyset$ for all $l > \bar{l}$. Consider a number $\bar{l}$ large enough such that

$$t_{\bar{l}} > r_n \qquad \text{and} \qquad t_{\bar{l}} > \frac{\text{OPT}(I)}{\alpha \min\{w_j | j \in I\}}.$$

Such $\bar{l}$ exists because the right hand side of both of the inequalities above are finite, and the sequence $\{t_l\}_{l=1}^{\infty}$ goes to infinity. The first constraint is to ensure that no request is released after time $t_{\bar{l}}$. Let us prove that the second constraint implies that all requests in $R_{\bar{l}}$ will be completed at the $\bar{l}^{\text{th}}$ iteration. If at least one request in $R_{\bar{l}}$ is not completed in the $\bar{l}^{\text{th}}$ iteration, we have the following inequality:

$$\frac{\beta}{\alpha}\text{OPT}(I) < \beta \min\{w_j | j \in I\}\, t_{\bar{l}} \leq \sum_{j \in R_{\bar{l}}} w_j f_{\alpha,\beta}(c_j^{\text{alg}_{\bar{l}}}, t_{\bar{l}}).$$

Because $\text{alg}_{\bar{l}}$ is the algorithm that minimizes the auxiliary offline problem, we have

$$\sum_{j \in R_{\bar{l}}} w_j f_{\alpha,\beta}(c_j^{\text{alg}_{\bar{l}}}, t_{\bar{l}}) \leq \sum_{j \in R_{\bar{l}}} w_j f_{\alpha,\beta}(c_j^{\text{OPT}}, t_{\bar{l}}).$$

Combining the two inequalities above, we obtain

$$\frac{\beta}{\alpha}\text{OPT}(I) < \sum_{j \in R_{\bar{l}}} w_j f_{\alpha,\beta}(c_j^{\text{OPT}}, t_{\bar{l}}) \leq \frac{\beta}{\alpha}\text{OPT}(I)$$

which is a contradiction. Therefore for all $l > \bar{l}$, $R_l = \emptyset$. □

Let us prove that constraints (3) and (4) are valid. We prove this by proving the following inequalities that follow from the definition of $\{C(r)\}_{r\in[0,\alpha]}$ and $\{T(r)\}_{r\in[0,\alpha]}$.

Lemma 2. *For $0 \leq r \leq r' \leq \alpha$,*

$$r(T(r) - T(r')) \leq C(r') - C(r) \leq r'(T(r) - T(r')).$$

Proof. By definition,

$$C(r') - C(r) = \sum_{l=1}^{\infty} \sum_{j \in A_l, rt_l < c_j^{\text{alg}_l} \leq r't_l} w_j c_j^{\text{alg}_l} \quad \text{and} \quad T(r) - T(r') = \sum_{l=1}^{\infty} \sum_{j \in A_l, rt_l < c_j^{\text{alg}_l} \leq r't_l} w_j t_l.$$

Clearly, for any $j \in A_l$ such that $rt_l < c_j^{\text{alg}_l} \leq r't_l$, $rt_l < c_j^{\text{alg}_l} \leq r't_l$. □

The following lemma shows that t_1 is small enough such that Condition 2 in Section 3.1 holds, which is essential for proving the lemmas in the rest of this section. The following lemma applies to all $\omega \in [0, 1)$, so it is valid for both $\text{PAC}_{\alpha,\beta}$ and $\text{RPAC}_{\alpha,\beta}$.

Lemma 3. *For both $PAC_{\alpha,\beta}$ and any realization of $RPAC_{\alpha,\beta}$, for any (offline or online) algorithm alg and any request j,*

$$c_j^{alg} > \frac{1}{1+2\alpha} t_1.$$

This lemma is used later with the algorithm alg being OPT and alg_1.

Proof. Because the server has a unit speed limit and a request cannot be visited before the release date, for any request j, $c_j^{\text{alg}} \geq \max(r_j, d(l_j, D))$. On the other hand, for all j, the t_1 determined by $\text{PAC}_{\alpha,\beta}$ is at most $\max(r_j, d(l_j, D))$. Therefore, for $\text{PAC}_{\alpha,\beta}$ and $\text{RPAC}_{\alpha,\beta}$ with $\omega = 0$, for all request j, $t_1 \leq c_j^{\text{alg}}$. For $\text{RPAC}_{\alpha,\beta}$ with any $\omega \in [0, 1)$, $t_1 < (1 + 2\alpha)c_i^{\text{alg}}$. □

The other linear inequalities require the following property of algorithm alg_l.

Lemma 4. *For any set S satisfying $A_l \subseteq S \subseteq \bigcup_{i=l}^{\infty} A_i$, alg_l is the offline algorithm that minimizes the cost:*

$$\sum_{j \in S} w_i f_{\alpha,\beta}\left(c_j^{alg}, t_l\right).$$

Proof. Let alg be any offline algorithm. Any request j in S but not R_l is released after t_l, and thus $c_j^{\text{alg}} \geq r_j \geq t_l$ and $f_{\alpha,\beta}\left(c_j^{\text{alg}}, t_l\right) = \beta t_l$. Therefore, we can decompose the cost into two terms as follows:

$$\sum_{j \in S} w_i f_{\alpha,\beta}\left(c_j^{\text{alg}}, t_l\right) = \sum_{j \in S \cap R_l} w_j f_{\alpha,\beta}\left(c_j^{\text{alg}}, t_l\right) + \sum_{j \in S \setminus R_l} w_j \beta t_l. \tag{9}$$

The second term is independent of the algorithm, so this lemma is equivalent to alg_l being optimal when the cost consists of the first term only.

Any request in R_l but not S is not in A_l, and thus is not completed by alg_l before time αt_l. For those requests j, we have $f_{\alpha,\beta}\left(c_j^{\text{alg}_l}, t_l\right) = \beta t_l$. Therefore, suppose on the contrary that algorithm alg_l' obtains a lower value for the first term on the right hand side of Equation (9) than alg_l, then

$$\begin{aligned}
&\sum_{j \in R_l} w_j f_{\alpha,\beta}\left(c_j^{\text{alg}_l}, t_l\right) \\
&= \sum_{j \in S \cap R_l} w_j f_{\alpha,\beta}\left(c_j^{\text{alg}_l}, t_l\right) + \sum_{j \in R_l \setminus S} w_j \beta t_l \\
&> \sum_{j \in S \cap R_l} w_j f_{\alpha,\beta}\left(c_j^{\text{alg}_l'}, t_l\right) + \sum_{j \in R_l \setminus S} w_j f_{\alpha,\beta}\left(c_j^{\text{alg}_l'}, t_l\right) \\
&= \sum_{j \in R_l} w_j f_{\alpha,\beta}\left(c_j^{\text{alg}_l'}, t_l\right),
\end{aligned}$$

which contradicts the definition of alg_l. □

Let us now prove that for $\beta > \alpha$, constraints (5) and (8) are valid for $\text{PAC}_{\alpha,\beta}$ and $\text{RPAC}_{\alpha,\beta}$ respectively. This can be proved by the following lemma that is obtained by using Lemma 4 with $S = \bigcup_{i=l}^{\infty} A_i$ and comparing the costs given by OPT and alg_l for each integer l.

Lemma 5. *When $\beta > \alpha$, for $PAC_{\alpha,\beta}$,*

$$OPT \geq \frac{2\alpha^2}{(1+2\alpha)(\beta-\alpha)} C(\alpha) + \frac{\alpha(\beta-\alpha(1+2\alpha))}{(1+2\alpha)(\beta-\alpha)} T(0), \tag{10}$$

and for $RPAC_{\alpha,\beta}$,

$$OPT \geq \frac{\alpha \ln(1+2\alpha)}{\beta-\alpha} C(\alpha) + \frac{(\beta-\alpha(1+2\alpha))\ln(1+2\alpha)}{2(\beta-\alpha)} T(0). \tag{11}$$

Proof. Consider Lemma 4 with $S = \bigcup_{i=l}^{\infty} A_i$. Since alg_l achieves the lowest cost value, we have

$$\sum_{j \in \bigcup_{i=l}^{\infty} A_i} w_j f_{\alpha,\beta}\left(c_j^{\text{OPT}}, t_l\right) \geq \sum_{j \in \bigcup_{i=l}^{\infty} A_i} w_j f_{\alpha,\beta}\left(c_j^{\text{alg}_l}, t_l\right). \tag{12}$$

For further discussion, we define $A_l^\star$ to be the set of requests that are completed in the time interval $(\alpha t_{l-1}, \alpha t_l]$ by OPT, that is, for all positive integers l,

$$A_l^\star \triangleq \left\{ j | j \in I, \alpha t_{l-1} < c_j^{\text{OPT}} \leq \alpha t_l \right\} \tag{13}$$

where $t_0 \triangleq \frac{t_1}{1+2\alpha}$ for simplicity. Note that for $\text{RPAC}_{\alpha,\beta}$, the sequence $\{t_l\}_{l=1}^{\infty}$ depends on the realization of ω, and thus the sets $\{A_l^\star\}_{l=1}^{\infty}$ are not deterministic. We drop the dependency on ω to simplify the notation. We define $\{A_l^\star\}_{l=1}^{\infty}$ this way in order to use the following inequalities:

$$f_{\alpha,\beta}\left(c_j^{\text{OPT}}, t_l\right) \leq \alpha t_l + \begin{cases} 0 & \text{for } c_j^{\text{OPT}} \leq \alpha t_l. \\ (\beta-\alpha) t_l & \text{for } c_j^{\text{OPT}} > \alpha t_l. \end{cases}$$

Using these inequalities, we have the following upper bound on the left hand side of (12):

$$\sum_{j \in \bigcup_{i=l}^{\infty} A_i} w_j f_{\alpha,\beta}\left(c_j^{\text{OPT}}, t_l\right) \leq \sum_{j \in \bigcup_{i=l}^{\infty} A_i} \alpha w_j t_l + \sum_{j \in \left(\bigcup_{i=l}^{\infty} A_i\right) \cap \left(\bigcup_{i=l+1}^{\infty} A_i^\star\right)} (\beta-\alpha) w_j t_l.$$

Because $\bigcup_{i=l+1}^{\infty} A_i^\star$ is a superset of $\left(\bigcup_{i=l}^{\infty} A_i\right) \cap \left(\bigcup_{i=l+1}^{\infty} A_i^\star\right)$, the inequality above still holds even if we replace the second term on the right hand side with the summation over the set $\bigcup_{j=l+1}^{\infty} A_j^\star$, which is what we will do.

On the other hand, we have the following equation for the right hand side of (12):

$$\sum_{j \in \bigcup_{i=l}^{\infty} A_i} w_j f_{\alpha,\beta}\left(c_j^{\text{alg}_l}, t_l\right) = \sum_{j \in A_l} w_j c_j^{\text{alg}_l} + \sum_{j \in \bigcup_{i=l+1}^{\infty} A_i} \beta w_j t_l.$$

Summing the above inequalities and equations over all positive integers l, we obtain

$$\sum_{l=1}^{\infty} \left(\sum_{j \in \bigcup_{i=l}^{\infty} A_i} \alpha w_j t_l + \sum_{j \in \bigcup_{i=l+1}^{\infty} A_i^\star} (\beta-\alpha) w_j t_l \right) \geq \sum_{l=1}^{\infty} \left(\sum_{j \in A_l} w_j c_j^{\text{alg}_l} + \sum_{j \in \bigcup_{i=l+1}^{\infty} A_i} \beta w_j t_l \right).$$

Interchanging the order of the summations on both sides, using equations $\sum_{l=1}^{i-1} t_l = \frac{t_i - t_1}{2\alpha}$, and arranging terms properly, the above inequality is equivalent to the following inequality:

$$\frac{\alpha}{1+2\alpha} \sum_{i=1}^{\infty} \sum_{j \in A_i^\star} w_j t_i \geq \frac{2\alpha^2}{(1+2\alpha)(\beta-\alpha)} C(\alpha)(I) + \frac{\alpha(\beta - \alpha(1+2\alpha))}{(1+2\alpha)(\beta-\alpha)} T(0)(I). \quad (14)$$

Since the left hand side of the above inequality is smaller than $\mathrm{OPT}(I)$, (10) holds.

For $\mathrm{RPAC}_{\alpha,\beta}$, since (14) is valid for any realization of ω, it is also valid when we take expectation on both sides. For each $j \in I$, the minimum value αt_l such that $c_j^{\mathrm{OPT}} \leq \alpha t_l$ has the following expected value:

$$\int_0^1 (1+2\alpha)^x c_j^{\mathrm{OPT}} \, \mathrm{d}x = \frac{2\alpha}{\ln(1+2\alpha)} c_j^{\mathrm{OPT}}. \quad (15)$$

This equation gives us the following equation:

$$\mathbb{E}\left(\sum_{i=1}^{\infty} \sum_{j \in A_i^\star} w_j t_i \right) = \sum_{j \in I} \frac{2}{\ln(1+2\alpha)} w_j c_k^{\mathrm{OPT}} = \frac{2}{\ln(1+2\alpha)} \mathrm{OPT}(I). \quad (16)$$

Taking expectation on both sides of (14), using (16), and rearranging terms properly, we obtain (11). □

Let us now prove Theorems 3 and 6 using the lemmas above.

Proof of Theorems 3 and 6. Consider Lemma 2 with $r = 0$ and $r' = \alpha$, we obtain

$$0 \geq C(\alpha) - \alpha T(0). \quad (17)$$

The inequality

$$\frac{(1+2\alpha)(1+\alpha)}{\alpha} \times (10) + \frac{\beta - (2\alpha^2 + 3\alpha)}{\beta - \alpha} \times (17)$$

proves Theorem 3. The inequality

$$\frac{2(1+\alpha)}{\ln(1+2\alpha)} \times (11) + \frac{\beta - (2\alpha^2 + 3\alpha)}{\beta - \alpha} \times (17)$$

proves Theorem 6. □

Let us now prove that constraint (6) is valid. This can be proved by the following lower bound on $\mathrm{OPT}(I)$ that is obtained by using Lemma 4 with $S = A_l \cup A_{l+1}$ and comparing the costs given by OPT and alg_l for each integer l.

Lemma 6.

$$OPT \geq \frac{\alpha}{2\beta} C(\alpha) + \frac{\alpha}{2+4\alpha} T(0). \quad (18)$$

Proof. Using Lemma 4 with $S = A_l \cup A_{l+1}$, we obtain

$$\sum_{j \in A_l \cup A_{l+1}} w_j f_{\alpha,\beta}\left(c_j^{\mathrm{OPT}}, t_l\right) \geq \sum_{j \in A_l \cup A_{l+1}} w_j f_{\alpha,\beta}\left(c_j^{\mathrm{alg}_l}, t_l\right) = \sum_{j \in A_l} w_j c_j^{\mathrm{alg}_l} + \sum_{j \in A_{l+1}} w_j \beta t_l. \tag{19}$$

The following inequality holds for any $x, y \geq 0$:

$$\frac{\beta}{\alpha} x \geq f_{\alpha,\beta}(x, y). \tag{20}$$

Combining (19) and (20), we obtain

$$\frac{\beta}{\alpha} \sum_{j \in A_l \cup A_{l+1}} w_j c_j^{\mathrm{OPT}} \geq \sum_{j \in A_l} w_j c_j^{\mathrm{alg}_l} + \sum_{j \in A_{l+1}} w_j \beta t_l.$$

Summing over all positive integers l, using Lemma 3 and the equation $t_{l+1} = (1+2\alpha)t_l$, we obtain the lemma. □

Lemma 6 and the fact that $C(\alpha)(I) + T(0)(I)$ is an upper bound on $\mathrm{PAC}_{\alpha,\beta}(I)$,

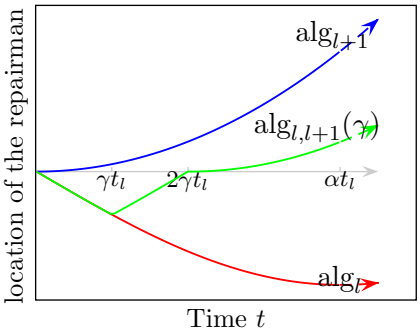
$$\max\left(\frac{2\beta}{\alpha}, 4 + \frac{2}{\alpha}\right)$$

is an upper bound on the competitive ratio of $\mathrm{PAC}_{\alpha,\beta}$. To obtain better bounds on the competitive ratio, we need to prove constraints (7), which is done by the following inequalities that is obtained by using Lemma 4 with $S = A_{l+1}$ and comparing alg_l with some algorithms that are combinations of alg_l and alg_{l+1} for each integer l.

Lemma 7. *For any* $\gamma \in \left[0, \frac{\alpha}{2}\right]$,

$$C(\alpha) + \frac{\beta - 2\gamma}{1+2\alpha} T(0) \leq C(\gamma) + \beta T(\gamma) + C\left(\frac{\alpha - 2\gamma}{1+2\alpha}\right) + \frac{\beta - 2\gamma}{1+2\alpha} T\left(\frac{\alpha - 2\gamma}{1+2\alpha}\right).$$

Figure 3: Illustration of $\mathrm{alg}_{l,l+1}(\gamma)$.



Proof. For each $\gamma \in [0, \frac{\alpha}{2}]$ and each positive integer l, we define the offline algorithm $\mathrm{alg}_{l,l+1}(\gamma)$ (Figure 3) as the following combination of algorithms alg_l and alg_{l+1}:

1. At time $0 \leq t \leq \gamma t_l$, the server follows the route of alg_l.
2. At time $\gamma t_l \leq t \leq 2\gamma t_l$, the server travels reversely so as to arrive at D by time $2\gamma t_l$.
3. Starting at time $2\gamma t_l$, the server follows a delayed version (delayed by $2\gamma t_l$) of alg_{l+1}.

Compare $\mathrm{alg}_{l,l+1}(\gamma)$ with alg_l in Lemma 4 with $S = A_l \cup A_{l+1}$, we obtain the following inequality:

$$\sum_{j \in A_l \cup A_{l+1}} w_j f_{\alpha,\beta}\left(c_j^{\mathrm{alg}_l}, t_l\right) \leq \sum_{j \in A_l \cup A_{l+1}} w_j f_{\alpha,\beta}\left(c_j^{\mathrm{alg}_{l,l+1}(\gamma)}, t_l\right). \tag{21}$$

The left hand side of (21) equals to

$$\sum_{j \in A_l} w_j c_j^{\mathrm{alg}_l} + \beta \sum_{j \in A_{l+1}} w_j t_l.$$

The part of the summation over A_l on the right hand side of (21) is

$$\sum_{j \in A_l, c_j^{\mathrm{alg}_l} \leq \gamma t_l} w_j c_j^{\mathrm{alg}_l} + \beta \sum_{j \in A_l, c_j^{\mathrm{alg}_l} > \gamma t_l} w_j t_l.$$

The part of the summation over A_{l+1} on the right hand side of Inequality (21) is at most

$$\frac{2\gamma}{1+2\alpha} \sum_{j \in A_{l+1}} w_j t_{l+1} + \sum_{j \in A_{l+1}, c_j^{\mathrm{alg}_{l+1}} \leq \frac{\alpha-2\gamma}{1+2\alpha} t_{l+1}} w_j c_j^{\mathrm{alg}_{l+1}} + \frac{\beta-2\gamma}{1+2\alpha} \sum_{j \in A_{l+1}, c_j^{\mathrm{alg}_{l+1}} > \frac{\alpha-2\gamma}{1+2\alpha} t_{l+1}} w_j t_{l+1}$$

where we have used the equation $t_{l+1} = (1+2\alpha)t_l$. This expression is not tight only if the server under $\mathrm{alg}_{l,l+1}(\gamma)$ happens to visit some locations of requests j in A_{l+1} earlier than $2\gamma t_l + c_j^{\mathrm{alg}_{l+1}}$.

Summing over all positive integers l and using Lemma 1 and 3, we obtain the lemma. □

Remark 5. *In our analysis, we first find inequalities related to terms of the forms*

$$\sum_{j \in A_l, c_j^{alg_l} \leq r t_l} w_j c_j^{alg_l} \qquad \textit{and} \qquad \sum_{j \in A_l, c_j^{alg_l} > r t_l} w_j t_l$$

and then we take the summation of the inequalities over all positive integers l to obtain linear inequalities as constraints. It may seem that we can obtain better bounds by considering each of the linear inequalities separately without taking the summation over all integers l. However, it is not the case. Denote $l_{\max}$ the maximum integer l such that $A_l \neq \infty$. We observe that the upper bounds on the competitive ratio of $PAC_{1,1}$ approach the objective value of $LP_{1,1}^{det}(N)$ when $l_{\max}$ increases. Similar results are also shown in the numerical analysis for the randomized online algorithms case. Therefore, we cannot reduce the upper bound on the competitive ratios significantly even if we did not take the summation over all positive integers l.

3.2.2 Lower Bounds on The Competitive Ratios

It is sufficient for us to prove the case where $\mathbb{M} = \mathbb{R}$ and $D = 0$.

Let us first consider the lower bounds for the deterministic algorithms. We first prove the first statement of Theorem 5. We start with the case in which $\alpha \in (0.5, 1]$ in the following lemma. After proving the lemma, we explain how we may modify the argument to show that the statement is true for all $\alpha \in (0, 1]$.

Lemma 8. *For $\alpha \in (0.5, 1]$, for each $\epsilon > 0$, define $I(\epsilon)$ to be the problem instance that contains only one request with $r_1 = 1$, $l_1 = (1 + 2\alpha)\alpha + \epsilon$, and $w_1 = 1$, then*

$$\lim_{\epsilon \to 0^+} \frac{PAC_{\alpha,\beta}(I(\epsilon))}{OPT(I(\epsilon))} \geq 3 + \frac{1}{\alpha}.$$

Proof. We first consider the cost of the optimal offline algorithm. One feasible offline algorithm is to let the server travel to the location of the request l_1 with full speed starting at time 0. The server arrives at time l_1, and $l_1 > r_1$ because we have assumed $\alpha > 0.5$. As a result,

$$\mathrm{OPT}(I(\epsilon)) \leq l_1 = (1 + 2\alpha)\alpha + \epsilon. \tag{22}$$

Now let us consider the cost of $\mathrm{PAC}_{\alpha,\beta}$. For this problem instance, $t_1 = r_1 = 1$, $t_2 = 1 + 2\alpha$, and $t_3 = (1 + 2\alpha)^2$. Because the completion time of the request in any offline algorithm is at least the location l_1, which is greater than αt_2, the request cannot be completed in the first two iterations. As a result, the cost of the online algorithm has the following lower bound:

$$\mathrm{PAC}_{\alpha,\beta}(I(\epsilon)) = c_1^{\mathrm{PAC}_{\alpha,\beta}} \geq t_3 + l_1 \geq (1 + 2\alpha)^2 + (1 + 2\alpha)\alpha + \epsilon > (1 + 2\alpha)(1 + 3\alpha). \tag{23}$$

Considering (22) and (23), the lemma holds. □

For fixed $r_1 = 1$, the key in the proof above is to design the location l_1 to be slightly greater than αt_l for some positive integer l (in the proof above, $l = 2$), with the additional constraint that $l_1 > 1 (= r_1)$. We can use the same idea to design adversarial problem instances when $\alpha \leq 0.5$. Defining k to be the minimum integer such that $(1 + 2\alpha)^k \alpha > 1$ and replacing the location of the only request by $(1 + 2\alpha)^k \alpha + \epsilon$, Lemma 8 holds for all $\alpha \in (0, 1]$. Therefore, Theorem 5 holds for all $\alpha \in (0, 1]$.

Now let us prove the second statement of Theorem 5. Similar to the proof of the previous lemma, we start with the case $\alpha > 0.5$, and then explain how we may generalize the proof for the case of all $\alpha \in (0, 1]$.

Lemma 9. *For $\alpha \in (0.5, 1]$, for each $\epsilon > 0$, define the problem instance $I(\epsilon)$ to have the following 3 requests.*

$$(r_j, l_j, w_j) \triangleq \begin{cases} (1, -((1 + 2\alpha)\alpha^2 - \epsilon), \epsilon^2) & \text{for } j = 1, \\ ((1 + 2\alpha)\alpha, (1 + 2\alpha)\alpha, \epsilon) & \text{for } j = 2, \\ ((1 + 2\alpha)\alpha + \epsilon, (1 + 2\alpha)\alpha + \epsilon, 1) & \text{for } j = 3, \end{cases}$$

then

$$\lim_{\epsilon \to 0^+} \lim_{\beta \to \infty} \frac{PAC_{\alpha,\beta}(I(\epsilon))}{OPT(I(\epsilon))} \geq \frac{(1 + \alpha)(1 + 2\alpha)}{\alpha}.$$

This lemma implies the following statement. For any real number $\rho < \frac{(1+\alpha)(1+2\alpha)}{\alpha}$, there exists an adversarial problem instance $I(\epsilon)$ on which the ratio between $\mathrm{PAC}_{\alpha,\beta}(I(\epsilon))$ and $\mathrm{OPT}(I(\epsilon))$ is greater than ρ for all big enough β. Thus, this lemma proves the second statement of Theorem 5 for the cases where $\alpha \in (0.5, 1]$.

Proof. The cost of the optimal algorithm is lower than the algorithm alg that travels to l_3 starting at time 0 and then travels to l_1 with maximum speed. Under algorithm alg, the completion time of Request 3 equals l_3, and as $\epsilon \to 0^+$, the completion time of any other request is bounded above by a constant, e.g., $\alpha(1+2\alpha)^2+1$. In addition, the total weight for the first two requests approaches 0 as $\epsilon \to 0^+$. As a result,

$$\mathrm{OPT}(I(\epsilon)) \le \mathrm{alg}(I(\epsilon)) = \sum_{j=1}^{2} w_j c_j^{\mathrm{alg}} + w_3 c_3^{\mathrm{alg}} \tag{24}$$

which approaches $(1+2\alpha)\alpha$ as $\epsilon \to 0^+$.

Now let us consider the online algorithm $\mathrm{PAC}_{\alpha,\beta}$. The key observation is that requests 1 and 3 are completed in the last iteration and Request 3 is visited after Request 1. Therefore, most of the weight (w_3) will have a large completion time.

The first request defines $t_1 = 1$, $t_2 = 1+2\alpha$, and $t_3 = (1+2\alpha)^2$. For a fixed α and a fixed problem instance $I(\epsilon)$, when β is big enough, $\mathrm{PAC}_{\alpha,\beta}$ maximizes the total weight of requests that can be completed at each iteration. Because $(1+2\alpha)\alpha^2 > \alpha$ when $\alpha > 0.5$, we can find a small enough ϵ such that $|l_1| > \alpha$. For such an ϵ, the first request cannot be completed in the first iteration, and therefore is in R_2. At time t_2, the auxiliary offline algorithm alg_2 will complete only the second request, because its weight ϵ is greater than that of the first request ϵ^2. As a result, the first request will not be completed in the second iteration, and thus is in R_3. Therefore, the set R_3 contains the first and the third requests.

At time t_3, the only way to complete both requests in R_3 is to visit l_1 first. Thus,

$$\lim_{\beta\to\infty} c_3^{\mathrm{PAC}_{\alpha,\beta}} \ge t_3 + 2|l_1| + |l_3| = (1+2\alpha)^2(1+\alpha) - \epsilon.$$

Therefore,

$$\lim_{\beta\to\infty} \mathrm{PAC}_{\alpha,\beta}(I(\epsilon)) \ge \lim_{\beta\to\infty} w_3 c_3^{\mathrm{PAC}_{\alpha,\beta}} = (1+2\alpha)^2(1+\alpha) - \epsilon. \tag{25}$$

Considering (24) and (25), the lemma holds. □

The key in the proof above is that the first request will not be completed until the last iteration, which makes the completion time of most of the weight large. For general $\alpha \in (0, 1]$, denote k as the smallest positive integer such that $(1+2\alpha)^k\alpha > 1$. For each $\epsilon > 0$, we define $I(\epsilon)$ to be the problem instance that has the following 3 requests:

$$(r_j, l_j, w_j) \triangleq \begin{cases} (1, -((1+2\alpha)^k\alpha^2 - \epsilon), \epsilon^2) & \text{for } j = 1. \\ ((1+2\alpha)^k\alpha, (1+2\alpha)^k\alpha, \epsilon) & \text{for } j = 2. \\ ((1+2\alpha)^k\alpha + \epsilon, (1+2\alpha)^k\alpha + \epsilon, 1) & \text{for } j = 3. \end{cases}$$

Using a similar argument as in the original proof, we can show that the lemma above holds for all $\alpha \in (0, 1]$. Thus, we have completed the proof of Theorem 5.

Let us now consider the lower bounds for $\mathrm{RPAC}_{\alpha,\beta}$. The first statement in Theorem 8 is implied by the following lemma.

Lemma 10. *Define I to be the problem instance that only has one request with $(r_1, l_1, w_1) = (1, 1, 1)$. Then,*

$$\frac{RPAC_{\alpha,\beta}(I)}{OPT(I)} = 1 + \frac{2\alpha}{\ln(1+2\alpha)}.$$

Proof. Clearly, $\mathrm{OPT}(I) = 1$.On the other hand, the expected cost of $\mathrm{RPAC}_{\alpha,\beta}$ is

$$\mathrm{RPAC}_{\alpha,\beta}(I) = 1 + \mathbb{E}(t_1) = l_1 + \int_0^1 (1+2\alpha)^x \,\mathrm{d}x = 1 + \frac{2\alpha}{\ln(1+2\alpha)}.$$

□

Similar to the comment between the statement and the proof of Lemma 9, the following lemma proves the second statement in Theorem 8.

Lemma 11. *For each $\epsilon > 0$, define $I(\epsilon)$ to be the problem instance with the following $2M+1$ requests:*

$$(r_j, l_j, w_j) \triangleq \begin{cases} (\epsilon, \epsilon^2 j, \epsilon^3) & \text{for } j = 1, \cdots, M \\ (\epsilon, -\epsilon^2 (j-M), \epsilon^6) & \text{for } j = M+1, \cdots, 2M \\ (\alpha, \alpha, 1) & \text{for } j = 2M+1 \end{cases}$$

where $M \triangleq \left\lfloor \frac{1}{\epsilon^2} \right\rfloor$. Then,

$$\lim_{\epsilon \to 0^+} \lim_{\beta \to \infty} \frac{RPAC_{\alpha,\beta}(I(\epsilon))}{OPT(I(\epsilon))} \geq \frac{2(1+\alpha)}{\ln(1+2\alpha)}.$$

Proof. The cost of the optimal algorithm is lower than the algorithm alg that travels to location 1 starting at time 0 and then travels to -1 with maximum speed. The completion time of request $(2M+1)$, which carries most of the weight, is α, and the completion time of all other requests is upper bounded by 3. Moreover, the total weight of the first $2M$ requests approaches 0 as $\epsilon \to 0^+$. As a result, as $\epsilon \to 0^+$, the cost of OPT has the following upper bound:

$$\mathrm{OPT}(I(\epsilon)) \leq \mathrm{alg}(I(\epsilon)) \to w_{2M+1} c_{2M+1}^{\mathrm{alg}} = \alpha. \tag{26}$$

Let us now consider the randomized algorithm $\mathrm{RPAC}_{\alpha,\beta}$. For a fixed α and a fixed problem instance $I(\epsilon)$, when β is large enough, the online algorithm maximizes the total weight of requests that can be completed at each iteration.

The problem instance $I(\epsilon)$ is designed so that when ϵ is small enough, the total weight of the $(M+1)^{\mathrm{th}}$ through the $2M^{\mathrm{th}}$ request is smaller than the weight of any other request. Therefore, none of the above mentioned M requests will be completed at iteration l if $t_l < 1$.

For each realization of ω, let k to be the smallest integer such that $t_k \geq 1$. The expected value of t_k is

$$\mathbb{E}(t_k) = \int_0^1 (1+2\alpha)^x \,\mathrm{d}x = \frac{2\alpha}{\ln(1+2\alpha)}.$$

At time t_k, the optimal offline auxiliary algorithm alg_k maximizes the number of requests that are completed among the $(M+1)^{\mathrm{th}}$ to the $2M^{\mathrm{th}}$ requests subject to the constraint

that the server arrives at location α by time αt_k. As a result, the completion time of the $(2M+1)^{\text{th}}$ request in any realization of ω has the following lower bound:

$$c_{2M+1}^{\text{RPAC}_{\alpha,\beta}} = t_k + c_{2M+1}^{\text{alg}_k} \geq (1+\alpha)t_k - 2\epsilon^2.$$

Consider this and the expected value of t_k calculated above, as $\epsilon \to 0^+$, we obtain the following inequality for the expected total weighted completion time of the randomized online algorithm:

$$\text{RPAC}_{\alpha,\beta}(I(\epsilon)) \geq \mathbb{E}\left(w_{2M+1} c_{2M+1}^{\text{RPAC}_{\alpha,\beta}}\right) \geq \mathbb{E}\left((1+\alpha)t_k - 2\epsilon^2\right) \to \frac{2\alpha(1+\alpha)}{\ln(1+2\alpha)}. \tag{27}$$

Considering (26) and (27), the lemma holds. □

3.3 Probabilistic Version

In this section, we consider the probabilistic version of the online WTRP described in Section 2.2.1, where assumptions 2, 3, and 4 are imposed. We prove that with these assumptions, the deterministic online algorithm ReOpt (formalized later) is almost surely asymptotically optimal. Finally, in Remark 6, we compare our results with that of Jaillet and Wagner [30].

The algorithm ReOpt is defined as the following:

1. If all released requests are completed, the server stays at its location.
2. When a request is released, the server calculates and follows the route so as to minimize the total weighted completion time of the released but not completed requests (starting at the server's current location).

Theorem 9. *For any sequence of problem instances $\{I^i\}_{i=1}^{\infty}$ where for any $i < n$, I^i is the first i requests in I^n, almost surely,*

$$\lim_{n\to\infty} \frac{ReOpt(I^n)}{OPT(I^n)} = 1.$$

In order to prove this theorem, we denote TSP_i the length of the shortest tour among the depot and the locations of the requests in I^i (TSP stands for the Traveling Salesman Problem). We need the following lemma that relates $ReOpt(I^n)$ with TSP_n:

Lemma 12.

$$ReOpt(I^n) \leq \sum_{j=1}^{n} w_j \left(r_j + O\left(\log n\right) TSP_n\right).$$

Proof of Theorem 9. For all $j \in [n]$, $c_j \geq r_j$. Thus, $OPT(I^n) \geq \sum_{j=1}^{n} w_j r_j$. With this inequality, Lemma 12, and Assumption 4, we have

$$\frac{ReOpt(I^n)}{OPT(I^n)} \leq 1 + \frac{\sum_{j=1}^{n} w_j O\left(\log n\right) \text{TSP}_n}{\sum_{j=1}^{n} w_j r_j} \leq 1 + \frac{n\Omega O(\log n) TSP_n}{\omega \sum_{j=1}^{n} r_j}.$$

According to Beardwood *et al.* [10] and Assumption 2, $TSP_n = O(n^{1-\frac{1}{M}})$ almost surely. On the other hand, according to Assumption 3, $\sum_{j=1}^{n} r_j$ can be written as $\sum_{i=1}^{n}(n +$

$1-i)Y_i$, which is $\Theta(n^2)$ almost surely according to the strong law of large numbers. Therefore, with probability one,

$$\frac{n\Omega O(\log n) TSP_n}{\omega \sum_{j=1}^{n} r_j} = \frac{\Omega}{\omega} O(n^{-\frac{1}{M}} \log n),$$

which approaches 0 as n goes to infinity. □

For further discussion, for any positive integer i, we denote ReOpt_i the route of the server under ReOpt when the problem instance consists of only the requests in I^i.

We prove Lemma 12 with the following two steps. First, we prove that for each positive integer i, the total weight of unserved requests decreases "exponentially" under ReOpt_i with a rate related to TSP_i:

Lemma 13. *For any integer $k \geq 0$, $1 \leq i \leq n$,*

$$\sum_{c_j^{ReOpt_i} > r_i + 3k\, TSP_i} w_i \leq 2^{-k} \sum_{j=1}^{i} w_i$$

where all summations are restricted to $1 \leq j \leq i$.

Second, we find an iterative relation for $\{\text{ReOpt}_i(I^i)\}_{i=1}^{\infty}$:

Lemma 14. *For any positive integers i and k,*

$$ReOpt_{i+1}(I^{i+1}) \leq ReOpt_i(I^i) + \left(r_{i+1} + \left(3k + \frac{3}{2}\right) TSP_{i+1}\right) w_{i+1} + \frac{3}{2} TSP_{i+1} 2^{-k} \sum_{j=1}^{i} w_j.$$

Proof of Lemma 12. According to Lemma 14 and mathematical induction, for any positive integer k,

$$\text{ReOpt}(I^n) = \text{ReOpt}_n(I^n) \leq \sum_{j=1}^{n} w_j \left(r_j + \left(3k + \frac{3}{2} + \frac{3}{2} n 2^{-k}\right) \text{TSP}_n\right).$$

Set $k = \lceil \log n \rceil$, the smallest integer that is at least $\log n$, we have

$$\begin{aligned} \text{ReOpt}(I^n) &\leq \sum_{j=1}^{n} w_j \left(r_j + \left(3\left(\log n + 1\right) + \frac{3}{2} + \frac{3}{2} n 2^{-\log n}\right) \text{TSP}_n\right) \\ &\leq \sum_{j=1}^{n} w_j \left(r_j + \left(3 \log n + 6\right) \text{TSP}_n\right) \end{aligned}$$

where we have used $n 2^{-\log n} = 1$. □

Proof of Lemma 13. For any $t \geq r_i$, we define an alternative route Alt_t (of ReOpt_i) as follows:

1. Follow ReOpt_i up to time t.
2. Return to the depot using the shortest path.

3. Follow the tour TSP_i.

The time it takes for the second step is at most $\frac{1}{2}\mathrm{TSP}_i$ and for the third step is TSP_i. Therefore, the completion time is at most $t + \frac{3}{2}\mathrm{TSP}_i$ for all requests (recall that in this lemma, we consider only requests in I^i.) Thus, the cost of the alternative route $\mathrm{Alt}_t(I^i)$ is at most

$$\mathrm{Alt}_t(I^i) \leq \sum_{c_j^{\mathrm{ReOpt}_i} \leq t} w_j c_j^{\mathrm{ReOpt}_i} + \sum_{c_j^{\mathrm{ReOpt}_i} > t} w_j \left(t + \frac{3}{2}\mathrm{TSP}_i \right)$$

where we restrict the summations to $j \in [i]$ through the proof of this lemma.

Because ReOpt_i achieves the lowest cost among all routes that are the same before time r_i, the cost of Alt_t for the first i requests is no smaller than that of ReOpt_i. Thus,

$$\begin{aligned}\sum_{c_j^{\mathrm{ReOpt}_i} > t} w_j \left(t + \frac{3}{2}\mathrm{TSP}_i \right) &\geq \sum_{c_j^{\mathrm{ReOpt}_i} > t} w_j c_j^{\mathrm{ReOpt}_i} \\ &\geq \sum_{t+3\mathrm{TSP}_i \geq c_j^{\mathrm{ReOpt}_i} > t} w_j t + \sum_{c_j^{\mathrm{ReOpt}_i} > t+3\mathrm{TSP}_i} w_j (t + 3\mathrm{TSP}_i).\end{aligned}$$

Canceling the same term $w_j t$ and dividing both sides by $3\mathrm{TSP}_i$, we obtain

$$\frac{1}{2} \sum_{c_j^{\mathrm{ReOpt}_i} > t} w_j \geq \sum_{c_j^{\mathrm{ReOpt}_i} > t+3\mathrm{TSP}_i} w_j.$$

This inequality, together with mathematical induction, completes the proof. □

Proof of Lemma 14. We consider the following alternative route Alt (of ReOpt_{i+1}):

1. Follow ReOpt_i until time $r_i + 3k\mathrm{TSP}_i$. (If this value is less than r_{i+1}, then go to step 2 at time r_{i+1}.)
2. Travel to the origin.
3. Follow the tour TSP_{i+1}.

We continue the proof with the case $r_i + 3k\mathrm{TSP}_i > r_{i+1}$. We can prove that the conclusion holds in the other case using a similar argument.

For all $j \in [i]$, if $c_j^{\mathrm{ReOpt}_i} \leq r_i + 3k\mathrm{TSP}_i$, then $c_j^{\mathrm{Alt}} = c_j^{\mathrm{ReOpt}_i}$; otherwise, $c_j^{\mathrm{ReOpt}_i} > r_i + 3k\mathrm{TSP}_i$ and thus

$$c_j^{\mathrm{Alt}} \leq r_i + 3k\mathrm{TSP}_i + \frac{1}{2}\mathrm{TSP}_i + \mathrm{TSP}_{i+1} < c_j^{\mathrm{ReOpt}_i} + \frac{3}{2}\mathrm{TSP}_{i+1}.$$

In addition, we have the following upper bound on the completion time of request $i+1$:

$$c_{i+1}^{\mathrm{Alt}} \leq r_i + 3k\mathrm{TSP}_i + \frac{1}{2}\mathrm{TSP}_i + \mathrm{TSP}_{i+1} \leq r_{i+1} + \left(3k + \frac{3}{2} \right) \mathrm{TSP}_{i+1}.$$

According to Lemma 13, the total weight of requests in I^i such that $c_j^{\mathrm{ReOpt}_i} > r_i + 3k\mathrm{TSP}_i$ is at most $2^{-k}\sum_{j=1}^{i} w_j$. As a result, we have

$$\mathrm{Alt}(I^{i+1}) \leq \mathrm{ReOpt}_i(I^i) + \left(r_{i+1} + \left(3k + \frac{3}{2}\right)\mathrm{TSP}_{i+1}\right) w_{i+1} + \frac{3}{2}\mathrm{TSP}_{i+1} 2^{-k} \sum_{j=1}^{i} w_j.$$

The rest follows from $\mathrm{ReOpt}_{i+1}(I^{i+1}) \leq \mathrm{Alt}(I^{i+1})$, which is the definition of ReOpt_{i+1}. □

Remark 6. *Theorem 9 is a special case of Theorem 3 in [30]. However, the proof of Theorem 3 in [30] is invalid as it contains a wrong claim (described in detail in the next paragraph). Using the arguments in this section, we can show that the general case of Theorem 3 in [30] does in fact hold.*

The proof of Theorem 3 in [30] relies on a lemma (Lemma 5 in [30]) whose proof wrongly assumes that

$$\max_{j\in\{1,2,\ldots,n-1\}} c_j^{ReOpt_{n-1}} \leq r_n + \frac{3}{2} TSP_{n-1}. \tag{28}$$

To see that this does not necessary hold, let us consider the problem instance that has the following 5 requests in the 2-dimensional Euclidean space:

$$(r_j, l_j, w_j) \triangleq \begin{cases} (0,(0,-10),1) & \text{for } j=1. \\ (0,(0,-9),10^4) & \text{for } j=2. \\ (0,(0,9),10^6) & \text{for } j=3. \\ (0,(0,10),10^2) & \text{for } j=4. \\ (1,(0,0),1) & \text{for } j=5. \end{cases}$$

When the first 4 requests are released, the algorithm $ReOpt_4$ travels in the order of locations l_3, l_2, l_4, and then l_1. Therefore, $\max_{j\in\{1,2,\ldots,n-1\}} c_j^{ReOpt_{n-1}} = 66$. On the other hand, the TSP tour for the first 4 requests has a length of 40. Therefore, $r_n + \frac{3}{2} TSP_{n-1} = 61 < 66$, which contradicts (28).

Here, we do not say whether Lemma 5 in [30] is true or not. Instead, we use an alternative lemma (Lemma 12) to prove that the main theorem holds.

4 Online Scheduling with Multi-state Machines

In this section, we develop online algorithms for the general case of online scheduling with multi-state machines as defined in Section 2.1, and based on the best (α, β) pair found in the previous section. For simplicity, we describe our algorithm and analysis for the basic setting defined in Section 2.1.1, but our results are valid for all problem variants described in Section 2.1.2. We assume $r_1 > 0$ to simplify the initialization stage of the algorithms and the corresponding analysis. However, it is clear that we can modify the algorithms and analysis without increasing the upper bounds on the competitive ratios for cases where $r_1 = 0$. As a minor caveat, we impose some mild technical assumptions as discussed in Section A.2.

In section 4.1, we propose a deterministic online algorithm Plan-and-Commit (PAC) and prove that the competitive ratio is upper bounded by 5.14. In section 4.2, for each $\alpha \leq 1$, we propose a randomized online algorithm Randomized α-Plan-and-Commit (RPAC_α). We show that the best possible algorithm in this class is RPAC_1, which has a competitive ratio of $4/\ln(3) \approx 3.64$.

4.1 Plan-and-Commit (PAC)

Algorithm

The algorithm has two stages: initialization and iterations.

The first stage starts at time 0 and ends at time r_1. During that stage, The algorithm PAC does not change the states of the machines and does not process any jobs. At time r_1, before finishing the initialization stage, PAC defines $\{t_l \triangleq r_1 3^{l-1}\}_{l=1}^{\infty}$, and $R_1 \triangleq K(I_{t_1})$.

The second stage consists of iterations. For each positive integer l, the l^{th} iteration starts at time t_l and ends at time t_{l+1}. At time t_l, PAC calculates what would an offline algorithm alg_l have done starting at time 0 so as to minimize the following cost:

$$\sum_{k \in R_l} h_k \left(\min\left(x_k, t_l\right)\right),$$

where R_l roughly represents all uncompleted projects that are active with respect to the partially revealed problem instance I_{t_l}. For $l = 1$, R_l is defined in the initialization stage; for $l \geq 2$, R_l is defined at the end of the $(l-1)^{\text{th}}$ iteration. Here we assume that such an offline algorithm alg_l exists, which is not necessarily true if the assumptions in Section A.2 is not imposed. See Section A for a detailed discussion.

From time t_l to time $2t_l$, PAC follows a delayed version (delayed by t_l) of algorithm alg_l with the following two modifications: First, if between time 0 and time t_l, alg_l processes some jobs that PAC has already completed before time t_l, then PAC does not process those jobs (but still control the states of the machines). Second, for each machine $i \in [m]$, PAC stops controlling the state of machine i and stops processing any jobs on machine i when the last job processed by machine i with completion time at most t_l under alg_l is completed. By doing so, PAC is not processing any job nor controlling any machine state at time $2t_l$. The algorithm PAC defines A_l to be the projects in R_l that are completed by time t_l under the offline algorithm alg_l, i.e., $A_l \triangleq \{k | k \in R_l, x_k^{\text{alg}_l} \leq t_l\}$. Following the definition, for all $k \in A_l$,

$$x_k^{\text{PAC}} \leq t_l + x_k^{\text{alg}_l}. \tag{29}$$

In addition for reasons mentioned for the online WTRP case, (29) is not necessarily tight because the jobs required for completing a project k can be completed jointly in multiple previous iterations.

From time $2t_l$ to time t_{l+1}, PAC controls the machine states so that all machines i are at their original states O_i at time t_{l+1}. This is feasible because the state spaces of the machines correspond to symmetric metric spaces. At time t_{l+1}, before entering the next iteration, PAC defines R_{l+1} to be R_l minus A_l plus the projects that become active between time t_l and t_{l+1}, i.e., $R_{l+1} \triangleq (R_l \setminus A_l) \cup (K(I_{t_{l+1}}) \setminus K(I_{t_l}))$.

Competitive Analysis

Theorem 10 with $N = 1200$ shows that PAC is 5.14-competitive. Similar to the online WTRP case, we have taken into account the numerical errors so the proof is rigorous.

Theorem 10. *For any positive integer N, the objective value of $LP^{det}(N)$ is an upper bound on the competitive ratio of PAC where $LP^{det}(N)$ is defined as follows:*

$$\max\ X_N + T_{0,N} \tag{30}$$

$$\text{s.t. } T_{i,j+1} + T_{i,j-1} \le 2T_{i,j} \qquad \text{for } \begin{cases} i = 0, \cdots, N \\ j = 1, \cdots, N-1 \end{cases} \tag{31}$$

$$X_{i+1} - X_i \ge T_{i,i} - T_{i+1,i} \qquad \text{for } i = 0, \cdots, N-1 \tag{32}$$

$$X_{i+1} - X_i \le T_{i,i+1} - T_{i+1,i+1} \qquad \text{for } i = 0, \cdots, N-1 \tag{33}$$

$$T_{0,\lfloor \frac{N}{3} \rfloor} + X_N \le 2. \tag{34}$$

$$X_N + T_{0,\lfloor \frac{N}{3} \rfloor} \le X_i + T_{i,N} + X_{\lceil \frac{N-2i}{3} \rceil} + T_{0,\lceil \frac{2i}{3} \rceil} + T_{\lfloor \frac{N-2i}{3} \rfloor, \lceil \frac{N-2i}{3} \rceil} \quad \text{for } i = 0, \cdots, \left\lfloor \frac{N}{2} \right\rfloor \tag{35}$$

$$X_i \ge 0 \qquad \text{for } i = 1, \cdots, N$$

$$T_{i,j} \ge 0 \qquad \text{for } \begin{cases} i = 0, \cdots, N \\ j = 0, \cdots, N-1 \end{cases}$$

$$X_0 = T_{i,0} = T_{N,i} = 0 \qquad \text{for } i = 0, 1, \cdots, N$$

The proof of this theorem is similar to that of Theorem 4 for the online WTRP. In the following proof, we omit arguments that are essentially the same as that of Theorem 4.

Proof. We first notice that using a similar argument as the proof of Lemma 1, we can show that $\{A_l\}_{l=1}^{\infty}$ forms a partition of active projects $K(I)$. Because of (29) and the concavity of h_k, the PAC(I) has the following upper bound:

$$\mathrm{PAC}(I) \le \sum_{l=1}^{\infty} \sum_{k \in A_l} h_k\left(x_k^{\mathrm{alg}_l}\right) + \sum_{l=1}^{\infty} \sum_{k \in A_l} h_k\left(t_l\right) \triangleq X(1)(I) + T(0,1)(I)$$

where

$$X(r)(I) \triangleq \sum_{l=1}^{\infty} \sum_{k \in A_l, x_k^{\mathrm{alg}_l} \le r t_l} h_k\left(x_k^{\mathrm{alg}_l}\right) \quad \text{and} \quad T(r,v)(I) \triangleq \sum_{l=1}^{\infty} \sum_{k \in A_l, x_k^{\mathrm{alg}_l} > r t_l} h_k\left(v t_l\right).$$

As a result,

$$\sup_I \frac{X(1)(I)}{\mathrm{OPT}(I)} + \frac{T(0,1)(I)}{\mathrm{OPT}(I)}$$

is an upper bound on the competitive ratio of PAC. We drop the parameter I and view $\{\frac{X(r)}{\mathrm{OPT}}\}_{r \in [0,1]}$ and $\{\frac{T(r,v)}{\mathrm{OPT}}\}_{r,v \in [0,1]}$ as variables in an LP. We then find linear inequalities

between $\{X(r)\}_{r\in[0,1]}$, $\{T(r,v)\}_{r,v\in[0,1]}$ and OPT that are valid for all problem instances I, and translate those inequalities into linear constraints in the LP. There are uncountably infinite many variables $X(r)$ and $T(r,v)$, so we cannot solve the LP numerically. Therefore, for all positive integers N, we define $\mathrm{LP}^{det}(N)$ by dividing $[0,1]$ into $N+1$ arithmetic steps, and for all $i=0,1,\ldots,N$, letting X_i to represent $X(\frac{i}{N})/OPT$ and for all $i,j=0,1,\ldots,N$, letting $T_{i,j}$ to represent $X(\frac{i}{N},\frac{j}{N})/OPT$. By definition, for all $i=0,1,\ldots,N$, $X_0=T_{i,0}=T_{N,i}=0$. Therefore, there is only a total of N^2+N nonnegative real variables $\{X_i\}_{i=1}^{N}$ and $\{T_{i,j}\}_{i=0\ j=1}^{N-1\ N}$.

Constraints (31) follows from the concavity of functions h_k. Constraints (32) and (33) follow from the definition of $\{T(r,v)\}_{r,v\in[0,1]}$ and $\{X(r)\}_{r\in[0,1]}$, and can be proven using an argument similar to the proof of Lemma 2. Similar to Lemma 3, the set A_1 is empty, and this fact is useful for proving the other constraints. Similar to Lemma 4, alg_l is the algorithm that minimizes the cost if the summation is taken over $A_l\cup A_{l+1}$, i.e., the cost

$$\sum_{k\in A_l\cup A_{l+1}} h_k\left(\min\left(x_k,t_l\right)\right). \tag{36}$$

Similar to Lemma 6, comparing cost (36) of alg_l and that of OPT, we obtain the following linear inequality:

$$T\left(0,\frac{1}{3}\right)+X(1)\le 2\mathrm{OPT}.$$

This inequality gives us (34). Similar to Lemma 7, for all integers l and all $r\in\left[0,\frac{1}{2}\right]$, define algorithm $\mathrm{alg}_{l,l+1}(r)$ to be the algorithm that does the following.

1. At time $0\le t\le rt_l$, follows algorithm alg_l.
2. At time $rt_l\le t\le 2rt_l$, control the machine states in a way such that the states of the machines i are O_i at time $2rt_l$.
3. Starting at time $2rt_l$, follow a delayed version (delayed by $2rt_l$) of algorithm alg_{l+1} with the two modifications that are also used in PAC.

Comparing cost (36) of alg_l and that of $\mathrm{alg}_{l,l+1}(r)$, we obtain

$$\begin{aligned}\sum_{k\in A_l} h_k\left(x_k^{\mathrm{alg}_l}\right)+\sum_{k\in A_{l+1}} h_k\left(t_l\right)\le &\sum_{k\in A_l,x_k^{\mathrm{alg}_l}\le rt_l} h_k\left(x_k^{\mathrm{alg}_l}\right)+\sum_{k\in A_l,x_k^{\mathrm{alg}_l}>rt_l} h_k\left(t_l\right)\\ &+\sum_{k\in A_{l+1},x_k^{\mathrm{alg}_{l+1}}\le(1-2r)t_l} h_k\left(2rt_l+x_k^{\mathrm{alg}_{l+1}}\right)\\ &+\sum_{k\in A_{l+1},x_k^{\mathrm{alg}_{l+1}}>(1-2r)t_l} h_k\left(t_l\right).\end{aligned}$$

Using two inequalities that comes from h_k being concave, $h_k\left(2rt_l+x_k^{\mathrm{alg}_{l+1}}\right)\le h_k\left(2rt_l\right)+h_k\left(x_k^{\mathrm{alg}_{l+1}}\right)$ and $h_k\left(t_l\right)\le h_k\left(2rt_l\right)+h_k\left((1-2r)t_l\right)$, and taking the summation over all integers l, we obtain the following inequality:

$$X(1)+T\left(0,\frac{1}{3}\right)\le X(r)+T\left(r,1\right)+T\left(0,\frac{2r}{3}\right)+X\left(\frac{1-2r}{3}\right)+T\left(\frac{1-2r}{3},\frac{1-2r}{3}\right).$$

This inequality gives us (35). □ □

4.2 Randomized α-Plan-and-Commit (RPAC$_\alpha$)

Algorithm

The randomized online algorithm RPAC$_\alpha$ has a single random variable ω, uniformly distributed in $[0,1)$. We use RPAC$_\alpha(\omega)$ to denote the algorithm with realization ω. For any $\omega \in [0,1)$, RPAC$_\alpha(\omega)$ has two major stages: initialization and iterations.

The first stage is initialization. In this stage, RPAC$_\alpha(\omega)$ does the same thing as PAC defined in Section 4.1 except that now $\{t_l \leftarrow (1+2\alpha)^{l-1+\omega} r_1\}_{l=1}^{\infty}$ and R_1 is defined at time t_1. We drop the dependency on ω when writing t_l for simplicity.

The second stage is composed of iterations. The l^{th} iteration begins at t_l and ends at t_{l+1}. At time t_l, RPAC$_\alpha(\omega)$ calculates the optimal offline algorithm alg$_l$ that would have minimized the following cost function:

$$\sum_{k \in R_l} \left(h_k\left(t_{l+1}\right) - h_k\left(t_l\right)\right) \delta\left(x_k > \alpha t_l\right) \tag{37}$$

where δ is the indicator function with value 1 if the argument of the function is a true statement and 0 otherwise throughout this paper; and R_l is defined either in the initialization stage (when $l = 1$) or the previous iteration (when $l \geq 2$). The existence of such an algorithm alg$_l$ is obvious because the cost (37) depends only on whether each x_k is greater than αt_l or not, which depends only on the set of jobs that are completed by time αt_l, and there are at most 2^n such sets. The intuition for choosing this cost is to minimize the increase of the original cost due to the projects that are not completed at each iteration.

From time t_l to $(1+\alpha)t_l$, RPAC$_\alpha(\omega)$ follows a delayed an modified version of alg$_l$, where the modifications are analogies of those for PAC described in Section 4.1. The algorithm RPAC$_\alpha(\omega)$ defines $A_l \triangleq \{k | k \in R_l, x_k^{\text{alg}_l} \leq t_l\}$. By doing so, for all $k \in A_l$,

$$x_k^{\text{RPAC}_\alpha(\omega)} \leq t_l + x_k^{\text{alg}_l}. \tag{38}$$

Between time $(1+\alpha)t_l$ and t_{l+1}, RPAC$_\alpha(\omega)$ controls the machine states such that all machines are at their original states at time t_{l+1}. At time t_{l+1}, before entering the next iteration, RPAC$_\alpha(\omega)$ defines $R_{l+1} \triangleq (R_l \setminus A_l) \cup (K(I_{t_{l+1}}) \setminus K(I_{t_l}))$.

Competitive Analysis

The main result is that the competitive ratio of RPAC$_1$ is $4/\ln(3) \approx 3.64$, which can be obtained from the following Theorem:

Theorem 11. *For all $\alpha \in (0,1]$, the competitive ratio of RPAC$_\alpha$ is $\frac{2(1+\alpha)}{\ln(1+2\alpha)}$.*

The proof of the theorem requires the following lemma that is an analogue of Lemma 4.

Lemma 15. *For all positive integers l and any realization $\omega \in [0,1)$:*

$$\begin{aligned}
&\sum_{k \in K(I)} \left(h_k\left(t_{l+1}\right) - h_k\left(t_l\right)\right) \delta\left(x_k^{RPAC_\alpha(\omega)} > (1+\alpha)t_l\right) \\
\leq &\sum_{k \in K(I)} \left(h_k\left(t_{l+1}\right) - h_k\left(t_l\right)\right) \delta\left(x_k^{OPT} > \alpha t_l\right).
\end{aligned}$$

Proof of Theorem 11. The lower bound is trivial. For the special case of the online WTRP, RPAC_α is reduced to $\mathrm{RPAC}_{\alpha,\infty}$. According to Theorem 6, the competitive ratio of RPAC_α is at least $\frac{2(1+\alpha)}{\ln(1+2\alpha)}$.

Now let us begin to prove the upper bound.

For a project k such that $(1+\alpha)t_{l+1} \geq x_k^{\mathrm{RPAC}_\alpha(\omega)} > (1+\alpha)t_l$, we have the upper bound on the cost incurred by the project: $h_k\left(x_k^{\mathrm{RPAC}_\alpha(\omega)}\right) \leq h_k\left((1+\alpha)t_{l+1}\right) \leq (1+\alpha)h_k\left(t_{l+1}\right)$. Taking the summation over all active projects k and rearranging terms, we have the following upper bound on $\mathrm{RPAC}_\alpha(\omega)(I)$:

$$\frac{\mathrm{RPAC}_\alpha(\omega)(I)}{1+\alpha} \leq \sum_{k\in K(I)} h_k(t_0) + \sum_{l=0}^{\infty}\sum_{k\in K(I)} \left(h_k\left(t_{l+1}\right) - h_k\left(t_l\right)\right)\delta\left(x_k^{\mathrm{RPAC}_\alpha(\omega)} > (1+\alpha)t_l\right)$$

where we have defined $t_0 \triangleq t_1/(1+2\alpha)$ for simplicity.

Using Lemma 15 and exchanging the order of the double summations, we obtain

$$\frac{\mathrm{RPAC}_\alpha(\omega)(I)}{1+\alpha} \leq \sum_{k\in K(I)}\sum_{l=0}^{\infty} h_k\left(t_{l+1}\right)\delta\left(\alpha t_{l+1} \geq x_k^{\mathrm{OPT}} > \alpha t_l\right). \tag{39}$$

Because h_k is concave and $h_k(0) = 0$, for $\alpha t_{l+1} \geq x_k^{\mathrm{OPT}}$, we have

$$\alpha h_k\left(t_{l+1}\right) \leq h_k\left(\alpha t_{l+1}\right) \leq \frac{\alpha t_{l+1}}{x_k^{\mathrm{OPT}}} h_k\left(x_k^{\mathrm{OPT}}\right).$$

Therefore, taking the expectation on both sides of (39), we obtain

$$\mathbb{E}\left(\mathrm{RPAC}_\alpha(\omega)(I)\right) \leq \frac{1+\alpha}{\alpha}\sum_{k\in K(I)}\left(\int_0^1 (1+2\alpha)^x \,\mathrm{d}x\right) h_k\left(x_k^{\mathrm{OPT}}\right) = \frac{2(1+\alpha)}{\ln(1+2\alpha)}\mathrm{OPT}(I). \tag{40}$$

□ □

Proof of Lemma 15. Obviously, for all $i \leq l-1$ and $k \in A_i$, $x_k^{\mathrm{RPAC}_\alpha(\omega)} < (1+\alpha)t_l$. On the other hand, for any project $k \in K(I) \setminus \bigcup_{i=1}^{l-1} A_i$ such that $x_k^{\mathrm{alg}_l} \leq \alpha t_l$, k is in A_l. According to (38), $x_k^{\mathrm{RPAC}_\alpha(\omega)} \leq t_l + x_k^{\mathrm{alg}_l} \leq (1+\alpha)t_l$. Therefore,

$$\begin{aligned}
&\sum_{k\in K(I)} \left(h_k\left(t_{l+1}\right) - h_k\left(t_l\right)\right)\delta\left(x_k^{\mathrm{RPAC}_\alpha(\omega)} > (1+\alpha)t_l\right) \\
\leq &\sum_{k\in K(I)\setminus\bigcup_{i=1}^{l-1} A_i} \left(h_k\left(t_{l+1}\right) - h_k\left(t_l\right)\right)\delta\left(x_k^{\mathrm{alg}_l} > \alpha t_l\right).
\end{aligned}$$

Similar to Lemma 4, alg_l satisfies the following inequality:

$$\begin{aligned}
&\sum_{k\in K(I)\setminus\bigcup_{i=1}^{l-1} A_i} \left(h_k\left(t_{l+1}\right) - h_k\left(t_l\right)\right)\delta\left(x_k^{\mathrm{alg}_l} > \alpha t_l\right) \\
\leq &\sum_{k\in K(I)\setminus\bigcup_{i=1}^{l-1} A_i} \left(h_k\left(t_{l+1}\right) - h_k\left(t_l\right)\right)\delta\left(x_k^{\mathrm{OPT}} > \alpha t_l\right).
\end{aligned} \tag{41}$$

Combining the two inequalities above, we obtain the lemma. □ □

Remark 7. *Here we consider the effect of alg_l being an approximation. For each pair of numbers $\rho \geq 1$ and $\gamma \leq 1$, we say that alg_l is a (ρ, γ)-approximation if the following inequality holds:*

$$\sum_{k \in R_l} \left(h_k\left(t_{l+1}\right) - h_k\left(t_l\right)\right) \delta\left(x_k^{alg_l} > \alpha t_l\right) \leq \rho \sup_{alg} \sum_{k \in R_l} \left(h_k\left(t_{l+1}\right) - h_k\left(t_l\right)\right) \delta\left(x_k^{alg} > \gamma \alpha t_l\right).$$

In other words, the cost of an (ρ, γ)-approximated algorithm is no more than ρ times of the optimal algorithm that operates with a shorter period (γ portion) of time.

If for all integers l, alg_l is replaced with a (ρ, γ)-approximation, then the competitive ratio of $RPAC_\alpha$ becomes $\frac{2(1+\alpha)\rho}{\ln(1+2\alpha)\gamma}$. Here the analysis goes through because when we are applying an analogue of Lemma 4 to prove (41), the selected set is a superset of R_l. For the deterministic version of this algorithm (where $\omega = 0$ with probability one), replacing $\int_0^1 (1+2\alpha)^x \, \mathrm{d}x$ with $1+2\alpha$ in (40), we obtain a competitive ratio of $\frac{(1+\alpha)(1+2\alpha)\rho}{\alpha\gamma}$. Among all $\alpha \in (0, 1]$, the minimum is achieved at $\alpha = \sqrt{2}$ with a competitive ratio of $(1+\sqrt{2})^2 \frac{\rho}{\gamma} \approx 5.83 \frac{\rho}{\gamma}$.

5 Concluding Remarks

In this paper, we have formulated a new class of online scheduling problems, the online scheduling problem with multi-state machines, which takes into account the case for which each machine has multiple states and the processing time of a job depends on the state of the machine. In addition, we formulated a new family of cost functions, which we named the total costs of active projects, that covers many practical cost functions such as the total weighted completion time and the makespan. For the general cases where the objective is to minimize the total costs of active projects in the online scheduling problem with multi-state machines, we derive a 5.14-competitive deterministic online algorithm PAC, and a 3.65-competitive randomized online algorithm RPAC_1. When applying these algorithms to the online WTRP, we obtain competitive ratios lower than the best in the literature. Finally, we proved that ReOpt is almost surely asymptotically optimal for the online WTRP.

Even though we have made progress in considering a novel class of online scheduling problems, some problems remain open. We conclude this paper by listing four such problems.

Open Problem 1. *For each $\alpha \in (0, 1]$ and $\beta \geq \alpha$, what is the competitive ratio of $PAC_{\alpha,\beta}$ and $RPAC_{\alpha,\beta}$ for the online WTRP? What is the competitive ratio of PAC for online scheduling with multi-state machines?*

This open problem could be tackled by either a tighter analysis of the algorithm or a better adversarial problem instance, or both.

Open Problem 2. *What is the competitive ratio of ReOpt for the online WTRP? What about for online scheduling with multi-state machines?*

We proved that ReOpt is almost surely asymptotically optimal. However, when not imposing stochastic assumptions, we do not even know whether ReOpt has a constant competitive ratio or not (for both the online WTRP and online scheduling with multi-state machines).

Open Problem 3. *What is the competitive ratios of the best-possible deterministic and randomized online algorithms for the online WTRP when the metric space is the non-negative real line, the real line, and the 2-dimensional Euclidean space?*

In the online VRPs, the difficulties in determining the best-possible competitive ratios typically arise when going from the non-negative real line to the real line, or when going from $1D$ (real line) to $2D$, while going from $2D$ to general metric spaces is usually straightforward.

Open Problem 4. *What is the competitive ratios of the best-possible deterministic and randomized online algorithms for the online WTRP for general metric spaces? What about for online scheduling with multi-state machines?*

This problem is probably the central one which we would like to get an answer to. The best-possible online algorithms could be ReOpt, $\text{PAC}_{\alpha,\beta}$ ($\text{RPAC}_{\alpha,\beta}$ for the randomized case), or other algorithms.

References

[1] Allahverdi, A., Gupta, J. N., and Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega*, 27(2):219–239.

[2] Allahverdi, A., Ng, C., Cheng, T., and Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985–1032.

[3] Anderson, E. J. and Potts, C. N. (2004). Online scheduling of a single machine to minimize total weighted completion time. *Mathematics of Operations Research*, 29(3):686–697.

[4] Aprea, M., Feuerstein, E., Sadovoy, G., and de Loma, A. (2009). Discrete online tsp. *Algorithmic Aspects in Information and Management*, pages 29–42.

[5] Archer, A. and Blasiak, A. (2010). Improved approximation algorithms for the minimum latency problem via prize-collecting strolls. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 429–447. Society for Industrial and Applied Mathematics.

[6] Ausiello, G., Bonifaci, V., and Laura, L. (2008). The online prize-collecting traveling salesman problem. *Inf.Process.Lett.*, 107(6):199–204.

[7] Ausiello, G., Demange, M., Laura, L., and Paschos, V. (2004). Algorithms for the on-line quota traveling salesman problem. *Inf.Process.Lett.*, 92(2):89–94.

[8] Ausiello, G., Feuerstein, E., Leonardi, S., Stougie, L., and Talamo, M. (2001). Algorithms for the on-line travelling salesman. *ALGORITHMICA*, 29:2001.

[9] Ausiello, G., Laura, L., and Pini, E. (2006). A diligent algorithm for ol-trp on the line. Technical report 03-06, Department of Computer and Systems Science, University of Rome "La Sapienza", Rome, Italy.

[10] Beardwood, J., Halton, J. H., and Hammersley, J. M. (1959). The shortest path through many points. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 55, pages 299–327. Cambridge Univ Press.

[11] Blom, M., Krumke, S. O., Paepe, W. E. D., and Stougie, L. (2001). The online tsp against fair adversaries. *INFORMS Journal on Computing*, 13:2001.

[12] Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, B., Raghavan, P., and Sudan, M. (1994). The minimum latency problem. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 163–171. ACM.

[13] Bonifaci, V., Lipmann, M., and Stougie, L. (2006). *Online multi-server dial-a-ride problems*. Aracne.

[14] Chakrabarti, S., Phillips, C., Schulz, A., Shmoys, D., Stein, C., and Wein, J. (1996). Improved scheduling algorithms for minsum criteria. *Automata, Languages and Programming*, pages 646–657.

[15] Chung, C., Nonner, T., and Souza, A. (2010). Srpt is 1.86-competitive for completion time scheduling. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1373–1388. Society for Industrial and Applied Mathematics.

[16] Correa, J. and Wagner, M. (2005). Lp-based online scheduling: From single to parallel machines. *Integer programming and combinatorial optimization*, pages 45–56.

[17] Feuerstein, E. and Stougie, L. (2001). On-line single-server dial-a-ride problems. *Theoretical Computer Science*, 268(1):91–105.

[18] Goel, G. and Mehta, A. (2008). Online budgeted matching in random input models with applications to adwords. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 982–991. Society for Industrial and Applied Mathematics.

[19] Goemans, M. and Kleinberg, J. (1998). An improved approximation ratio for the minimum latency problem. *Mathematical Programming*, 82(1):111–124.

[20] Goemans, M. X. (1997). Improved approximation algorthims for scheduling with release dates. In *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, pages 591–598. Society for Industrial and Applied Mathematics.

[21] Goemans, M. X., Queyranne, M., Schulz, A. S., Skutella, M., and Wang, Y. (2002). Single machine scheduling with release dates. *SIAM Journal on Discrete Mathematics*, 15(2):165–192.

[22] Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, 5(2):287–326.

[23] Günther, E., Maurer, O., Megow, N., and Wiese, A. (2013). A new approach to online scheduling: Approximating the optimal competitive ratio. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 118–128. SIAM.

[24] Hall, L. A., Schulz, A. S., Shmoys, D. B., and Wein, J. (1997). Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Mathematics of Operations Research*, 22(3):513–544.

[25] Hall, L. A., Shmoys, D. B., and Wein, J. (1996). Scheduling to minimize average completion time: off-line and on-line algorithms. In *Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms*, SODA '96, pages 142–151, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

[26] Jaillet, P. and Lu, X. (2009). Online traveling salesman problems with flexibility. In Barnhart, C., Clausen, U., Lauther, U., and Møhring, R. H., editors, *Models and Algorithms for Optimization in Logistics*, Dagstuhl Seminar Proceedings, Dagstuhl, Germany. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany. Keywords: Online TSP, service flexibility, rejection options.

[27] Jaillet, P. and Lu, X. (2011). Online traveling salesman problems with service flexibility. *Networks*, 58(2):137–146.

[28] Jaillet, P. and Wagner, M. R. (2006). Online routing problems: Value of advanced information as improved competitive ratios. *Transportation Science*, 40(2):200–210.

[29] Jaillet, P. and Wagner, M. R. (2008). Online vehicle routing problems: A survey. *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 221–237.

[30] Jaillet, P. and Wagner, M. R. (2010). Almost sure asymptotic optimality for online routing and machine scheduling problems. *Networks*, 55(1):2–12.

[31] Jain, K., Mahdian, M., Markakis, E., Saberi, A., and Vazirani, V. V. (2003). Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *Journal of the ACM (JACM)*, 50(6):795–824.

[32] Jain, K., Mahdian, M., and Saberi, A. (2002). A new greedy approach for facility location problems. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 731–740. ACM.

[33] Kim, D.-W., Kim, K.-H., Jang, W., and Chen, F. F. (2002). Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer-Integrated Manufacturing*, 18(3):223–231.

[34] Kim, S. and Bobrowski, P. (1994). Impact of sequence-dependent setup time on job shop scheduling performance. *THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH*, 32(7):1503–1520.

[35] Koutsoupias, E. and Papadimitriou, C. H. (1995). On the k-server conjecture. *Journal of the ACM (JACM)*, 42(5):971–983.

[36] Krumke, S. O., de Paepe, W. E., Poensgen, D., and Stougie, L. (2003). News from the online traveling repairman. *Theoretical Computer Science*, 295(1):279–294.

[37] Lipmann., M. (2003). *On-line Routing*. PhD thesis, Technische Universiteit Eindhoven.

[38] Liu, P. and Lu, X. (2009). On-line scheduling of parallel machines to minimize total completion times. *Computers and Operations Research*, 36(9):2647–2652.

[39] Mahdian, M., Nazerzadeh, H., and Saberi, A. (2007). Allocating online advertisement space with unreliable estimates. In *Proceedings of the 8th ACM conference on Electronic commerce*, pages 288–294. ACM.

[40] Mahdian, M. and Yan, Q. (2011). Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pages 597–606. ACM.

[41] Mahdian, M., Ye, Y., and Zhang, J. (2002). *Improved approximation algorithms for metric facility location problems*, pages 229–242. Approximation algorithms for combinatorial optimization. Springer.

[42] Megow, N. and Schulz, A. S. (2004). On-line scheduling to minimize average completion time revisited. *Operations Research Letters*, 32(5):485–490.

[43] Megow, N., Uetz, M., and Vredeveld, T. (2006). Models and algorithms for stochastic online scheduling. *Mathematics of Operations Research*, 31(3):513–525.

[44] Mirrokni, V. S., Gharan, S. O., and Zadimoghaddam, M. (2012). Simultaneous approximations for adversarial and stochastic online budgeted allocation. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1690–1701. SIAM.

[45] Nowicki, E. and Zdrzałka, S. (1990). A survey of results for sequencing problems with controllable processing times. *Discrete Applied Mathematics*, 26(2):271–287.

[46] Phillips, C., Stein, C., and Wein, J. (1995). Scheduling jobs that arrive over time. *Algorithms and Data Structures*, pages 86–97.

[47] Pinedo, M. L. (2012). *Scheduling: theory, algorithms, and systems*. Springer.

[48] Shabtay, D. and Steiner, G. (2007). A survey of scheduling with controllable processing times. *Discrete Applied Mathematics*, 155(13):1643–1666.

[49] Shmoys, D. B., Wein, J., and Williamson, D. P. (1995). Scheduling parallel machines on-line. *SIAM Journal on Computing*, 24(6):1313–1331.

[50] Vinod, V. and Sridharan, R. (2008). Dynamic job-shop scheduling with sequence-dependent setup times: simulation modeling and analysis. *The International Journal of Advanced Manufacturing Technology*, 36(3):355–372.

[51] Yi, Y. and Wang, D. (2003). Soft computing for scheduling with batch setup times and earliness-tardiness penalties on parallel machines. *Journal of Intelligent Manufacturing*, 14(3):311–322.

A On the Existence of alg_l

In this section, we discuss the existence of alg_l. In Section A.1, we show that alg_l does not always exist by providing an example. In Section A.2, we provide a sufficient condition under which alg_l must exist.

A.1 A Nonexistence Example

Assume $m = 1$, $\mathbb{M}_1 = \mathbb{N}_{\geq 1}$, $O_1 = 1$, and $d_1(i, j) = 1$ for all $i \neq j$. Assume the cost is the total unweighted completion time (hence $h_k(x) = x$). Let there be only one job with $r_1 = 1$, and

$$p_{11}(i) = \begin{cases} \infty & \text{if } i = 1 \\ 1 + \frac{1}{i} & \text{if } i \geq 2 \end{cases}.$$

In the first iteration ($l = 1$), the job cannot be completed. In the second iteration ($l = 2$), the goal is to minimize $h_1(\min(x_1, t_2))$, which equals $\min(c_1, 3)$. If processed in state i, $i \in \mathbb{N}_{\geq 1} \setminus \{1\}$, the best possible completion time of job 1 is $2 + 1/i$ (1 for directing the state from 1 to i, $1 + 1/i$ for the processing time). On the other hand, if processed in State 1, the completion time is infinity. Hence, the infimum of the cost is 2, but is not achievable by any offline algorithm. Therefore, such an offline algorithm alg_2 does not exist.

A.2 A Sufficient Condition for the Existence of alg_l

In this section, we show that the optimal offline algorithm alg_l exists when two mild technical assumptions are imposed.

The first assumption is about the metric space:

Assumption 5 (Compactness)**.** *For any machine $i \in [m]$ and any $k \in \mathbb{R}_{\geq 0}$, the subset of the metric space $\{s : s \in \mathbb{M}_i, d_i(O_i, s) \leq k\}$ is compact.*

This assumption is valid in many metric spaces of practical use such as a closed subset of a multi-dimensional Euclidean space and a discrete metric space such that the number of points within any finite distance from O_i is finite. This assumption is violated for some artificially designed metric spaces such as the one discussed in Section A.1.

The second assumption is regarding the processing time:

Assumption 6 (Lower-semicontinuity)**.** *For any $i \in [m]$ and $j \in [n]$, p_{ij} is lower-semicontinuous.*

This assumption is valid in many practical settings such as the online vehicle routing problems and the case where all processing time functions are continuous. However, for artificially designed functions that violate this assumption, alg_l does not necessarily exist. For example, assume $m = 1$, $\mathbb{M}_1 = [0, 1]$, $d_1(i, j) = |i - j|$, and $O_1 = 0$. Assume the cost is the total unweighted completion time (hence $h_k(x) = x$). Let there be only one job with $r_1 = 1$, and

$$p_{11}(s) = \begin{cases} \infty & \text{for } s = 0.5 \\ |s - 0.5| & \text{for } s \neq 0.5. \end{cases}$$

Because the processing time is always non-zero, $x_1 = c_1 > r_1 = t_1$. Therefore, the job is not completed in the first iteration. In the second iteration, the goal is to minimize $h_1(\min(x_1, 3))$, which equals $\min(c_1, 3)$. When being processed in any states $s \neq 0.5$. the optimal completion time for the job is $1 + |s - 0.5|$ (1 for the release date and $|s - 0.5|$ for the processing time.) Therefore, the infimum of $\min(x_1, t_2)$ equals 1, but is not achievable by any offline algorithm.

The primary result in this appendix is the following lemma:

Lemma 16. *With Assumptions 5 and 6, there exists an (offline) algorithm that minimizes the following cost:*

$$\sum_{k \in R_l} h_k \left(\min \left(x_k, t_l \right) \right).$$

Proof. Let $\{\text{alg}^i\}_{i=1}^{\infty}$ be a sequence of algorithms such that

$$\lim_{i \to \infty} \sum_{k \in R_l} h_k \left(\min \left(x_k^{\text{alg}^i}, t_l \right) \right) = \inf_{\text{alg}} \sum_{k \in R_l} h_k \left(\min \left(x_k^{\text{alg}}, t_l \right) \right)$$

where two algorithms alg^i and $\text{alg}^{i'}$ are allowed to be the same even if $i \neq i'$.

Now let us consider the following sequence of n-dimensional real vectors: $S \triangleq \{(\min(c_1^{\text{alg}^i}, t_l), \min(c_2^{\text{alg}^i}, t_l), \ldots, \min(c_n^{\text{alg}^i}, t_l))\}_{i=1}^{\infty}$. Since the vectors are in a compact subset of the n-dimensional Euclidean space ($[0, t_l]^n$), there exists a limit point. Without loss of generality, we assume S converges to a limit point, and denote this limit point $(v_1, v_2, \ldots, v_n)$, i.e., for all $j \in [n]$, set $v_j \triangleq \lim_{i \to \infty} \min(c_j^{\text{alg}^i}, t_l)$. It is sufficient to prove that there is an algorithm alg' such that for all job $j \in [n]$, $\min(c_j^{\text{alg}'}, t_l) \leq v_j$ because for all $k \in R_l$, $\min(x_k, t_l)$ is a non-decreasing function of $\{\min(c_j, t_l)\}_{j=1}^n$.

Let us now construct such an algorithm alg'. If $v_j = t_l$, then we are not concerned about the completion time of job j under alg' because $\min(c_j, t_l) \leq t_l$ independent of the algorithm. Therefore, for simplicity, we can assume for all $j \in [n]$, $v_j < t_l$, and all jobs are completed by one of the machines under algorithm alg^i (for all large enough i). With this assumption, there exists a subsequence of algorithms $\{\text{alg}^i\}_{i=1}^{\infty}$ such that each job is processed by the same machine among the algorithms because there are a finite number (m^n) of possible combinations. Without loss of generality, we assume $\{\text{alg}^i\}_{i=1}^{\infty}$ is itself such a subsequence. Now we consider jobs processed by each machine separately. For simplicity, we say that all jobs are processed by Machine 1. We denote ρ a permutation of $[n]$ such that $\{v_{\rho(j)}\}_{j=1}^n$ is non-decreasing. For all positive integers i and jobs $j \in [n]$, we denote $s^{j,i}$ the state of Machine 1 under which job $\rho(j)$ is processed under algorithm alg^i. Because of the compactness (5) assumption, the sequence $\{(s^{1,i}, s^{2,i}, \ldots, s^{n,i})\}_{i=1}^{\infty}$ has a limit point $(\bar{s}^1, \bar{s}^2, \ldots, \bar{s}^n)$ where for all $j \in [n]$, $\bar{s}^j \in S_1$. We let alg' to process jobs in the order of $\rho(1), \rho(2), \ldots, \rho(n)$ at states $\bar{s}^1, \bar{s}^2, \ldots, \bar{s}^n$, respectively.

It is sufficient to prove that for all $j \in [n]$, $c_{\rho(j)}^{\text{alg}'} \leq \lim_{i \to \infty} c_{\rho(j)}^{\text{alg}^i}$. To prove this, first note that $c_{\rho(j)}^{\text{alg}'} = \sum_{i=1}^{j} d(\bar{s}^{j-1}, \bar{s}^j) + p_{1\rho(j)}(\bar{s}^j)$ and $c_{\rho(j)}^{\text{alg}^i} \geq \sum_{i=1}^{j} d(s^{i,j-1}, s^{i,j}) + p_{1\rho(j)}(s^{i,j})$, where $\bar{s}^0 \triangleq O_1$ and for all i, $s^{i,0} \triangleq O_1$ for simplicity. The conclusion follows directly from the fact $\{s^{j,i}\}_{i=1}^{\infty}$ converges to $\bar{s}^j$ and the lower-semicontinuity (6) assumption. $\square$