Online Scheduling with Multi-State Machines[*]

Dawsen Hwang[‡] Patrick Jaillet[§]

August 2015; revised July 2016, March 2017, November 2017

Abstract

In this paper, we propose a general framework for online scheduling problems in which each machine has multiple states that lead to different processing times. For these problems, in addition to deciding 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 include a complete proof 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.

Keywords: online algorithms; competitive analysis; machine scheduling; multi-state; traveling repairman; deterministic and randomized algorithms

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 to minimize a given cost function of all job completion times. *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 supremum of the 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.

One limitation in the majority of the scheduling literature (in particular when dealing with the online version of the problems) 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

[*]Accepted for publications in *Networks*. Research funded in part by AFOSR grant FA9550-10-1-0437, and by ONR grants N00014-12-1-0033 and N00014-15-1-2083

[‡]Google, Chicago, IL 60607; dawsen@google.com. Research done while at the Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, 02139; dawsen@mit.edu

[§]Department of Electrical Engineering and Computer Science, Operations Research Center, MIT, Cambridge, MA, 02139; jaillet@mit.edu times. This is the case for many manufacturing problems such as producing paper bags, semiconductors, and automobiles [2, 28, 42, 46]. For example, Pinedo [42] describes an application in a paper bag factory, where a machine has different states, each corresponding to the size of bags 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

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 state. In addition, we introduce a family of generic cost functions for these problems that can describe several practical objectives such as minimizing the total weighted completion time and the quota-collecting makespan. 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*. 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 (PAC and $RPAC_1$) 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. We address 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 (5.14 for PAC and 3.65 for $RPAC_1$) are smaller than the existing ones for the online WTRP (a special case in the general framework) in both deterministic and randomized settings (where the best existing results are 5.83 and 3.87, respectively).

Third, for a probabilistic version of the online WTRP, we provide a complete proof that the 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.

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 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 many different versions of scheduling problems. We adopt the conventional three-fold notation $\alpha|\beta|\gamma$ first introduced in [19] for describing some of the classical scheduling problems. The α field represents the machine environment, the β field represents the processing characteristics and constraints, and the γ field represents the cost function. For example $1|r_j|\sum w_j c_j$ refers to the problem of minimizing the total weighted completion time on one machine subject to release-date constraints, and $P|r_j|\sum w_j c_j$ is the corresponding problem when there are m identical parallel machines. The reader is referred to [19] and [42] 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 [40, 43]). 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 [29] and Vinod and Sridharan [45] 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, 12, 13, 16, 18, 21, 33, 37, 38, 41, 44]. The goal in this line of research is to derive online algorithms with the smallest competitive ratios. Recently, Günther *et al.* [20] develop an algorithmic approach that can approximate the best-possible competitive ratios for many versions of these problems. We note that the solution techniques described in most of the papers cited above cannot be adapted to fit our problems because the effect of the machine states on the processing time is not taken into account. The only possible exception is [21] whose ideas can be used to design algorithms for our problems (see the discussion below in the solution techniques). However, our approach involves other design ideas which allow us to get improved competitive ratios.

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 quota-collecting makespan, total weighted completion time, etc.), and underlying metric spaces (the non-negative real line, the real line, general continuous metric spaces, discrete metric spaces, etc.) result in different problems. The reader is referred to [24] 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 nomadic version of the online quota traveling salesman problem [5, 6, 22], and the latency online dial-a-ride problem (with infinite capacity) [11, 14, 32] (see Remark 2.3 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 latency online dial-a-ride problem 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 [14] 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 [32]. Krumke *et al.* [31] 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 [23] and Krumke *et al.* [31] propose two different deterministic online algorithms with a competitive ratio of 5.83 for general metric spaces. The randomized versions 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) [23, 31]. For the online latency online dial-a-ride problem with capacity 1, Feuerstein and Stougie [14] 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 requests that have not been served whenever a new request is released. The algorithm *ReOpt* belongs to the class of *zealous algorithms*, defined by Blom *et al.* [9], in which the server always travels with the maximum speed when there are requests that have not been served 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.* [7] 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 [25] give the only known performance guarantee for *ReOpt*. They 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, the proof of this result relies on the validity of a statement (Lemma 5 in [25]) whose proof is problematic and cannot easily be fixed. Instead, we propose in this paper an alternative complete proof.

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 [23] and Krumke *et al.* [31] 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 [21]. 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 approximation algorithms for the (offline) TRP [10, 17].

Instead of the actual completion time, the above auxiliary problem only considers whether a request is served by 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.* [21] 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.* [21] 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 [30] use this idea in the design of the well-known ($2k-1$)-competitive Work Function Algorithm (WFA) 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 [4, 15, 26, 27, 34, 35, 36, 39], its application to our problems remains challenging because of the particular choice of suitable variables, objectives, and constraints for factor-revealing LPs strategies to become successful.

1.3 Organization

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 several classical ones such as 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 new 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. We cannot prove that the analysis is tight as the best lower bounds we have obtained 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 provide an alternative complete proof that *ReOpt* is almost surely asymptotically optimal as the number of requests approaches infinity.

In Section 4, we consider general online scheduling problems 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 as in [23, 31] cannot easily 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*. In addition, we introduce a generic class of cost functions, which we call *the total costs of active projects*. In Section 2.2, we formally describe the online WTRP and show that the online WTRP is a special case of the new class of scheduling problems.

2.1 Online Scheduling with Multi-State Machines

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 different across problem instances. 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 for the 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, denoted by c_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.
- In the offline version of the problem, the number of jobs n and the characteristics of all the jobs are known ahead of time. In the online version, the number of jobs n is not known ahead of time and the characteristics of job j are revealed to the online algorithms at its release date r_j.

Cost Functions: We introduce a generic class of cost functions, which we call the *total costs of active projects*. Before describing its mathematical formulation, we first introduce the intuition behind it and 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; the project is said to be *active* when the jobs released so far include a subset of jobs whose characteristics collectively satisfy those conditions. Only active projects are counted in the overall cost. The cost of an active project is a function of the completion time of that project (i.e., the earliest time when all jobs in one of the sets 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 $\rho_j \in P$ gives the characteristics of job j other than r_j and $\{p_{ij}\}_{i=1}^{m}$).

This 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, which can be labeled as $\mathbb{N}$, are involved, where project $j \in \mathbb{N}$ 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, the notation $K(I)$ denotes 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 by x_k, is defined to be the earliest time at which all jobs in one of the satisfying sets 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 quantity given by the MinMax operator 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

$$\text{cost}(I) \triangleq \sum_{k \in K(I)} h_k(x_k).$$

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

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

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

For our results to hold, we need to impose the following assumption on the cost functions:

Assumption 2.1. *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 three in Table 1. 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 $\delta > 0$. For discussion on the necessity of imposing such an assumption, see Appendix B.

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., $I_t \triangleq \{(r_j, \{p_{ij}\}_{i=1}^m, \rho_j) | r_j \leq t\}$. The goal is to design online algorithms with strong performance in term of competitive analysis without consideration of computational complexity.

Online algorithms can be further classified as *deterministic* or *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 randomizes over a collection of deterministic online algorithms.

For notational convenience, for a deterministic (online or offline) algorithm called ALG, we denote by 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 the problem instance I under the deterministic algorithm ALG as

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

For a randomized online algorithm called ALG, defined by a collection of deterministic algorithms $\{ALG(\omega)\}$ where ω is drawn from a probability distribution Δ, we denote $ALG(I)$ as the expected cost under the problem instance I, i.e., $ALG(I) \triangleq \mathbb{E}_{\omega \sim \Delta}(ALG(\omega)(I))$. We drop I and ω when it is clear from context.

For each problem instance I, we define $OPT(I)$ as the infimum of the costs among all algorithms, i.e.,

$$OPT(I) \triangleq \inf_{ALG} ALG(I).$$

When the infimum is attainable, $OPT(I)$ corresponds to the cost of an optimal offline algorithm. Here we use the infimum rather than the minimum in the expression of $OPT(I)$ 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 A.3.

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

Definition 2.2 (Competitive Analysis)**.** *An online algorithm ALG is c-competitive, $c \in \mathbb{R}_{\geq 1}$, if for any problem instance I,*

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

The competitive ratio *of ALG is the infimum of c such that ALG is c-competitive.*

2.2 Online Weighted Traveling Repairman Problem

In this section we show how the online Weighted Traveling Repairman Problem (WTRP) can be seen as a special case of the general framework described in Section 2.1.

Let us first describe the online WTRP in precise terms. 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 request j is the earliest time greater than or equal to r'_j at which the server arrives at l_j (the on-site service time is zero). The objective is to minimize the total weighted completion time.

Let us now show how we can formulate the online WTRP as a scheduling problem within 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 2.3. The release date and the processing time in the above formulation are applicable to many variants of the VRPs. For example, when the cost function is the quota-collecting makespan as described in Table 1, the problem becomes the nomadic version of the online quota traveling salesman problem [5, 6, 22]. If we allow precedence constraints in the problem formulation, our general framework can cover the latency online dial-a-ride problems (with infinite capacity) [11, 14, 32]. 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, labeled as $2j-1$ and $2j$, where $r_{2j-1} = r_{2j} = r'_j$,

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

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

3 The Online Weighted Traveling Repairman Problem

In this section, we study the online WTRP as formulated in Section 2.2. More precisely, in Section 3.1, we propose and analyze a family of deterministic online algorithms. In Section 3.2, we consider the randomized versions of these algorithms. In Section 3.3, we study a probabilistic version of the online WTRP, and provide a proof that *ReOpt* is almost surely asymptotically optimal.

3.1 (α, β)-Plan-and-Commit

In Section 3.1.1, we propose a family of algorithms, parameterized by a pair of numbers (α, β) satisfying $\alpha \in (0, 1]$ and $\beta \in [\alpha, \infty)$. In Section 3.1.2, we analyze the proposed algorithms and provide upper and lower bounds on the competitive ratios, and determine the parameters (α, β) that lead to the smallest provable competitive ratio.

3.1.1 The Algorithms

For each pair of real numbers (α, β) such that $\alpha \in (0, 1]$ and $\beta \in [\alpha, \infty)$, we define a deterministic online algorithm (α, β)-Plan-and-Commit ($PAC_{\alpha,\beta}$) as follows:

Algorithm 1 (α, β)-Plan-and-Commit Algorithms ($PAC_{\alpha,\beta}$) for the online WTRP

1. Initialization

   The server stays at the depot D during the entire initialization phase.

   (a) At time $t = 0$, set $\tau \leftarrow \min\{d(D, l_j) | j \in I_0, l_j \neq D\}$. By convention, if $\{j \in I_0, l_j \neq D\}$ is an empty set, then set $\tau \leftarrow \infty$.

   (b) At time $t \in (0, \tau)$, if a request is revealed, then set $t_1 \leftarrow t$ and end the initialization phase.

   (c) At time $t = \tau$, set $t_1 \leftarrow \tau$ and end the initialization phase. (If both $\{j \in I_0, l_j \neq D\}$ and $I \setminus I_0$ are empty sets, then τ remains ∞ and there will be no end to the initialization phase.)

   At the time when the initialization phase ends: For all positive integer l, set $t_l \leftarrow t_1 \times (1 + 2\alpha)^{l-1}$. Define $R_1 \triangleq I_{t_1}$.

2. Repeat for $l = 1, 2, \dots,$

   (a) **Plan:** At time $t = t_l$, calculate what an offline algorithm, ALG_l, would have done between time 0 and αt_l to minimize $\sum\limits_{j \in R_l} w_j f_{\alpha,\beta}(c_j, t_l)$.

   Denote by A_l the set of requests in R_l completed by ALG_l before time αt_l.

   (b) **Commit:**

   - During time $t \in [t_l, (1 + \alpha)t_l]$, follow a delayed version of the route of ALG_l (delayed by t_l).
   - During time $t \in [(1+\alpha)t_l, (1+2\alpha)t_l]$, return to the depot D by traveling through the reverse of the route of ALG_l.

   (c) At time t_{l+1}, define $R_{l+1} \triangleq (R_l \setminus A_l) \cup (I_{t_{l+1}} \setminus I_{t_l})$.

The algorithm $PAC_{\alpha,\beta}$ has two major phases: initialization and iterations.

Initialization Phase: The server stays at the depot D during the entire duration of this phase. The phase begins at time 0 and, assuming it ends, transitions to the second phase at a time t_1, which is a time variable computed by the algorithm $PAC_{\alpha,\beta}$. Let us explain how our algorithms compute t_1. 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 $\{t_l \triangleq t_1(1 + 2\alpha)^{l-1}\}_{l=1}^{\infty}$ is 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 online or offline algorithm (including OPT) except for the simple 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 $PAC_{\alpha,\beta}$ that will end up being t_1 when the initialization ends. At time $t = 0$, $PAC_{\alpha,\beta}$ 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 which are away from the depot. 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 $PAC_{\alpha,\beta}$ sets $t_1 = \tau$. Otherwise, $PAC_{\alpha,\beta}$ sets t_1 to be the earliest release date of such requests. Note that in the case where all the requests are at the depot and are with release dates 0, the initialization phase will never end. However, in this case, the completion time of all requests will be 0 for all online and offline algorithms, and hence the cost will be 0 for both the optimal offline algorithm and $PAC_{\alpha,\beta}$. Therefore, such problem instances have no impact on the competitive ratio of the algorithm and we can ignore them when analyzing the competitive ratio.

Clearly, Condition 1 is satisfied from the way we have defined t_1. To see that Condition 2 is also satisfied, we note that the server has a unit speed limit. Therefore, request j cannot be visited before time $d(l_j, D)$. Furthermore, request j cannot be visited before its release date r_j. Thus, for any online or offline algorithm ALG, $c_j^{ALG} \geq \max(r_j, d(l_j, D))$. On the other hand, the variable t_1 determined by $PAC_{\alpha,\beta}$ is at most $\min_j \max(r_j, d(l_j, D))$. Therefore, for any online or offline algorithm ALG, $t_1/(1+2\alpha) < t_1 \leq c_j^{ALG}$.

Before starting the first iteration of the second phase at time t_1, we define R_1 to be the set of all requests with release date at most t_1, i.e., $R_1 \triangleq I_{t_1}$, and we define $t_l = t_1 \times (1 + 2\alpha)^{l-1}$ for all $l \geq 1$.

Iterations Phase: For any $l \geq 1$, the l^{th} iteration begins at time t_l and ends at time t_{l+1}, i.e., $[t_l, t_{l+1}]$. The algorithm is called Plan-and-Commit because it *plans* the route at time t_l and is *committed* to follow that route until time t_{l+1}, regardless of the requests released between time t_l and t_{l+1}.

At time t_l, we define an *auxiliary offline problem* as follows:

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

where

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

and the set R_l is defined either in the initialization phase (if $l = 1$) or in the previous iteration (if $l \geq 2$). It is clear that an optimal solution exists for this auxiliary problem whose objective value can be attained by an algorithm. This is because the number of permutations of jobs in R_l is finite, and for each permutation (giving the order the requests are served) the best way to serve them is to travel between them along shortest paths, and if necessary, wait at the location of a request until it is released before traveling to the next one. Call the optimal algorithm ALG_l, and let $c_j^{ALG_l}$, $j \in R_l$, be the corresponding optimal completion times. Note that ALG_l is an algorithm that starts at time 0. Also, even though all optimal completion times are well defined by ALG_l, for any job j for which $c_j^{ALG_l} > \alpha t_l$, the completion time is replaced by a penalty term βt_l in the objective value of the auxiliary problem.

In other words, one can think of αt_l as a deadline for the auxiliary problem. For each job in R_l whose completion will exceed αt_l, and so will be part of the set R_{l+1} in the auxiliary offline problem associated with the next iteration (remember that $t_{l+1} = (1 + 2\alpha)t_l$), then its contribution to the objective function at iteration l is instead associated with the term βt_l.

In the extreme case when $\beta \to \infty$, the auxiliary offline problem puts an emphasis on requests that can be completed before the next time step. The other parameter α represents a tradeoff between using more information in the auxiliary offline problem and having a smaller ratio between successive time steps. To see this, we note that the route of the optimal solution in the auxiliary offline problem is only related to requests revealed up to time αt_l. Therefore, when α is larger, we use more information for the design of ALG_l. However, when α is larger, the deadline αt_l is also larger, and the online algorithm needs to spend more time to visit the requests served by ALG_l up to time αt_l.

The online 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 by A_l those requests in R_l that have a completion time no greater than αt_l under the offline algorithm ALG_l. Following the definition of A_l, for all j in A_l, the completion time of our algorithm, $c_j^{PAC_{\alpha,\beta}}$, satisfies

$$c_j^{PAC_{\alpha,\beta}} \leq t_l + c_j^{ALG_l}. \tag{1}$$

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., $R_{l+1} \triangleq (R_l \backslash A_l) \cup (I_{t_{l+1}} \backslash I_{t_l})$,[1] which is guaranteed to be a superset of all released and not visited requests at time t_{l+1}.

We close the description of the algorithm by discussing three possible alternative options one could imagine for the design of ALG_l, and their impacts on the results we have obtained on competitive analysis, as described in Section 3.1.2.

1. Approximating the auxiliary problem: If ALG_l is an approximate solution to the auxiliary offline problem, then, in general, the analysis in Section 3.1.2 fails because Lemma 3.9 requires ALG_l to be an exact optimal solution.

2. Replacing the release dates of requests in R_l by 0: Since the online algorithm follows a delayed version of ALG_l, it may appear that the release dates of requests in R_l are not relevant and could be replaced with 0 when solving the auxiliary offline problem. However, doing so will break our analysis, in particular, Lemma 3.12. One intuition for why it is important to keep the release dates of requests in R_l when designing ALG_l goes as follows. The release date of a request is a lower bound on the completion time of the request in any online or offline algorithm, so the optimal offline algorithm has a "tendency" to serve requests with smaller release dates earlier. By keeping the release dates of requests in R_l, ALG_l and $PAC_{\alpha,\beta}$ would also have a "tendency" to serve requests with smaller release dates earlier, and thus become more like the optimal offline algorithm, which is useful for achieving a lower competitive ratio. Removing the release dates while designing ALG_l would eliminate this useful feature.

[1]The sets $I_{t_{l+1}}$ and I_{t_l} are derived from the definitions of I_t, t_{l+1}, and t_l.

3. Re-optimizing after computing A_l: At each iteration l, if we replace ALG_l by an alternative algorithm ALG'_l 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 to the depot D at time t_{l+1}, then the upper bounds on the competitive ratios are still valid because the total cost of the online algorithm corresponding to $\{ALG'_l\}_{l=1}^{\infty}$ is at most $\sum_{l=1}^{\infty} \sum_{j \in A_l} w_j \left(t_l + c_j^{ALG'_l}\right)$, which is at most $\sum_{l=1}^{\infty} \sum_{j \in A_l} w_j \left(t_l + c_j^{ALG_l}\right)$.

3.1.2 Competitive Analysis

We first describe our results. The detailed proofs regarding the upper bounds and lower bounds appear in Subsections 3.1.2.1 and 3.1.2.2, respectively.

Upper Bounds: For general pairs of (α, β) satisfying $\alpha \in (0, 1]$ and $\beta \geq \alpha$, the lowest upper bounds that we prove are related to the following linear program, which is parameterized by a positive integer N:

For $\beta = \alpha$

$$\underset{(C_1, C_2, \ldots C_N, T_0, T_1, \ldots, T_{N-1}) \in \mathbb{R}_{\geq 0}^{2N}}{\text{maximize}} \quad C_N + T_0 \qquad (\mathrm{LP}_{\alpha,\beta}^{det}(N))$$

subject to

$$C_{i+1} - C_i \geq \frac{i\alpha}{N}(T_i - T_{i+1}) \qquad \text{for } i = 0, \ldots, N-1 \qquad (2a)$$

$$C_{i+1} - C_i \leq \frac{(i+1)\alpha}{N}(T_i - T_{i+1}) \qquad \text{for } i = 0, \ldots, N-1 \qquad (2b)$$

$$1 \geq \frac{\alpha}{2\beta} C_N + \frac{\alpha}{2 + 4\alpha} T_0 \qquad (2c)$$

$$C_N + \frac{\beta - 2\frac{i\alpha}{N}}{1 + 2\alpha} T_0 \leq C_i + \beta T_i + C_{\lceil \frac{N-2i}{1+2\alpha} \rceil} + \frac{\beta - 2\frac{i\alpha}{N}}{1 + 2\alpha} T_{\lfloor \frac{N-2i}{1+2\alpha} \rfloor} \quad \text{for } i = 0, \ldots, \left\lfloor \frac{N}{2} \right\rfloor \qquad (2d)$$

where C_0 and T_N are defined to be 0.
For $\beta > \alpha$

$$\underset{(C_1, C_2, \ldots C_N, T_0, T_1, \ldots, T_{N-1}) \in \mathbb{R}_{\geq 0}^{2N}}{\text{maximize}} \quad C_N + T_0 \qquad (\mathrm{LP}_{\alpha,\beta}^{det}(N))$$

subject to

$$(2a), (2b), (2c), (2d)$$

$$1 \geq \frac{2\alpha^2}{(1 + 2\alpha)(\beta - \alpha)} C_N + \frac{\alpha(\beta - \alpha(1 + 2\alpha))}{(1 + 2\alpha)(\beta - \alpha)} T_0 \ (\text{if } \beta > \alpha) \qquad (2e)$$

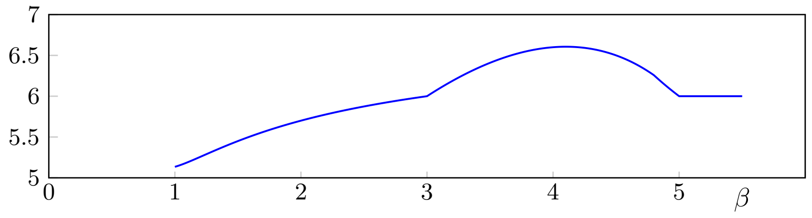
where C_0 and T_N are defined to be 0.

The main result can be stated as follows:

Theorem 3.1. *Let (α, β) be a pair of real numbers satisfying $\alpha \in (0,1]$ and $\beta \geq \alpha$. For any positive integer N, the objective value of $LP^{det}_{\alpha,\beta}(N)$ is an upper bound on the competitive ratio of $PAC_{\alpha,\beta}$.*

Theorem 3.1 allows us to select parameters α and β such that the upper bound on the competitive ratio is minimized. We numerically calculate the optimal objective value for all pairs of α, β when they are both multiples of 0.01 by using the software package Gurobi. As an example, Figure 1 illustrates the optimal objective values of $\text{LP}^{det}_{\alpha,\beta}(N)$ with $N = 10000$ and $\alpha = 1$ for different values of β.

Figure 1: Optimal objective values of $\text{LP}^{det}_{\alpha,\beta}(N)$ with $N = 10000$ and $\alpha = 1$ for different values of β



Our observation of all our 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 believe that $(\alpha, \beta) = (1, 1)$ is the parameter that minimizes the upper bound provided by Theorem 3.1. For $PAC_{1,1}$, we have the following corollary:

Corollary 3.2. *The competitive ratio of $PAC_{1,1}$ is at most* 5.14.

Proof. According to Gurobi, the objective value of $\text{LP}^{det}_{\alpha,\beta}(N)$ with $N = 10000$ and $\alpha = \beta = 1$ is slightly below 5.135, and the maximum possible numerical error is smaller than 0.005. Therefore, using Theorem 3.1, we conclude that the competitive ratio of $PAC_{1,1}$ is at most 5.14. □

The next corollary (Corollary 3.3) is weaker than Corollary 3.2 but still shows that the competitive ratio of $PAC_{1,1}$ is lower than the smallest competitive ratio in the literature (5.83). We include Corollary 3.3 because we can analytically prove this corollary (proved in Appendix C).

Corollary 3.3. *The competitive ratio of $PAC_{1,1}$ is at most* $39/7 \approx 5.57$.

In addition to the case of $\alpha = \beta = 1$, our observation of the numerical results indicates that as $N \to \infty$, when we fix $\alpha \in (0,1]$, for all $\beta \geq 2\alpha^2 + 3\alpha$, the optimal objective values of $\text{LP}^{det}_{\alpha,\beta}(N)$ are the same. For example, in Figure 1, for all $\beta \geq 5$, the optimal objective values are the same. In fact, for any $\alpha \in (0,1]$, when $\beta \geq 2\alpha^2 + 3\alpha$, our best provable upper bounds on the competitive ratios have a closed-form expression, as formalized in the following proposition:

Proposition 3.4. *For any $\alpha \in (0,1]$, when $\beta \geq 2\alpha^2 + 3\alpha$, the competitive ratio of $PAC_{\alpha,\beta}$ is at most $\frac{(1+2\alpha)(1+\alpha)}{\alpha}$.*

Lower Bounds: For lower bounds, we have the following main result:

Theorem 3.5. *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 3.5 together with Proposition 3.4 show that the analysis is tight when $\beta \to \infty$. Furthermore, for the algorithm that achieves the best provable upper bound on the competitive ratio, $PAC_{1,1}$, we have the following corollary for its lower bound:

Corollary 3.6. *The competitive ratio of $PAC_{1,1}$ is at least 4.*

Proof. Theorem 3.5 with $\alpha = 1$ proves this corollary. □

3.1.2.1 Upper Bounds on the Competitive Ratios

We first note that, for the sake of determining the upper bounds on the competitive ratios, 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 $l_j = D$. We first prove Theorem 3.1 and then Proposition 3.4.

For proving Theorem 3.1, we need to introduce a series of lemmas. First, we show that $\{A_l\}_{l=1}^{\infty}$ forms a partition of all requests I:

Lemma 3.7. *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 $I = \bigcup_{l=1}^{\infty} A_l$.

It is clear that $I \supset \bigcup_{l=1}^{\infty} A_l$. Therefore, it is sufficient to prove that $I \subset \bigcup_{l=1}^{\infty} A_l$, which is equivalent to the existence of an integer $\bar{l}$ such that $I = \bigcup_{l=1}^{\bar{l}} A_l$. Consider a number $\bar{l}$ large enough such that

$$t_{\bar{l}-(n-1)} > r_n \qquad \text{and} \qquad t_{\bar{l}} - (n-1) > \max\{d(D, l_j) | j \in I\}.$$

At each iteration l such that $l \geq (\bar{l} - (n-1))$, either A_l contains at least one request or all requests in I must have been completed. Since there are a total of n requests, all requests must have been completed by the $\bar{l}^{\text{th}}$ iteration. As a result, $I \subset \bigcup_{l=1}^{\infty} A_l$, which completes the proof. □

Because of Lemma 3.7 and (1), for any problem instance I, $PAC_{\alpha,\beta}(I)$ has the following upper bound:

$$PAC_{\alpha,\beta}(I) \leq \sum_{l=1}^{\infty} \sum_{j \in A_l} w_j c_j^{ALG_l} + \sum_{l=1}^{\infty} \sum_{j \in A_l} w_j t_l = C(\alpha)(I) + T(0)(I), \tag{3}$$

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

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

As a result, $\sup_I \frac{C(\alpha)(I)}{OPT(I)} + \frac{T(0)(I)}{OPT(I)}$ is an upper bound on the competitive ratio of $PAC_{\alpha,\beta}$. In order to find a small upper bound on $\sup_I \frac{C(\alpha)(I)}{OPT(I)} + \frac{T(0)(I)}{OPT(I)}$, we find inequalities between $\{T(r)(I)\}_{r\in[0,\alpha]}$, $\{C(r)(I)\}_{r\in[0,\alpha]}$ and $OPT(I)$ that are valid for all problem instances I. For notational convenience, we drop the parameter I when it is clear from context.

The first sets of inequalities are a direct result of the definition of $\{T(r)\}_{r\in[0,\alpha]}$ and $\{C(r)\}_{r\in[0,\alpha]}$ and are independent of the online algorithm:

Lemma 3.8. *For $0 \le r \le r' \le \alpha$,*

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

Proof. Clearly, for any $j \in A_l$ such that $rt_l < c_j^{ALG_l} \le r't_l$, we have $c_j^{ALG_l} \le r't_l$. Therefore,

$$\sum_{l=1}^{\infty} \sum_{j\in A_l, rt_l < c_j^{ALG_l} \le r't_l} w_j c_j^{ALG_l} \le \sum_{l=1}^{\infty} \sum_{j\in A_l, rt_l < c_j^{ALG_l} \le r't_l} w_j r' t_l = r' \sum_{l=1}^{\infty} \sum_{j\in A_l, rt_l < c_j^{ALG_l} \le r't_l} w_j t_l,$$

which gives $C(r') - C(r) \le r'(T(r) - T(r'))$. Similarly, for any $j \in A_l$ such that $rt_l < c_j^{ALG_l} \le r't_l$, we have $c_j^{ALG_l} > rt_l$. Therefore,

$$\sum_{l=1}^{\infty} \sum_{j\in A_l, rt_l < c_j^{ALG_l} \le r't_l} w_j c_j^{ALG_l} > \sum_{l=1}^{\infty} \sum_{j\in A_l, rt_l < c_j^{ALG_l} \le r't_l} w_j r t_l = r \sum_{l=1}^{\infty} \sum_{j\in A_l, rt_l < c_j^{ALG_l} \le r't_l} w_j t_l,$$

which gives $C(r') - C(r) \ge r(T(r) - T(r'))$. □

The rest of the inequalities are based on the fact that for all positive integers l, ALG_l is an optimal solution to the auxiliary offline problem. In addition, the following lemma shows that ALG_l is also optimal if the summation of the cost in the auxiliary offline problem was taken over any set of requests satisfying some properties:

Lemma 3.9. *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:*

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

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

$$\sum_{j\in S} w_i f_{\alpha,\beta}\left(c_j^{ALG}, t_l\right) = \sum_{j\in S\cap R_l} w_j f_{\alpha,\beta}\left(c_j^{ALG}, t_l\right) + \sum_{j\in S\setminus R_l} w_j \beta t_l. \quad (4)$$

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 in S is not in A_l, and thus is not completed by ALG_l before time αt_l. For those requests j, we have $f_{\alpha,\beta}\left(c_j^{ALG_l}, t_l\right) = \beta t_l$. Therefore, suppose to the contrary that algorithm ALG_l' obtains a lower value for the first term on the right hand side of Equation (4) than ALG_l. Then

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

which contradicts the definition of ALG_l. □

Comparing the costs given by OPT and ALG_l for each integer l by using Lemma 3.9 with $S = \bigcup_{i=l}^{\infty} A_i$, we obtain the following lemma:

Lemma 3.10. *When $\beta > \alpha$,*

$$OPT \geq \frac{2\alpha^2}{(1+2\alpha)(\beta-\alpha)} C(\alpha) + \frac{\alpha(\beta - \alpha(1+2\alpha))}{(1+2\alpha)(\beta-\alpha)} T(0). \tag{5}$$

Proof. Consider Lemma 3.9 with $S = \bigcup_{i=l}^{\infty} A_i$. Since ALG_l achieves the lowest cost value, we have

$$\sum_{j\in\bigcup_{i=l}^{\infty} A_i} w_j f_{\alpha,\beta}\left(c_j^{OPT}, t_l\right) \geq \sum_{j\in\bigcup_{i=l}^{\infty} A_i} w_j f_{\alpha,\beta}\left(c_j^{ALG_l}, t_l\right). \tag{6}$$

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,

$$A_l^\star \triangleq \left\{ j | j \in I, \alpha t_{l-1} < c_j^{OPT} \leq \alpha t_l \right\} \tag{7}$$

where $t_0 \triangleq \frac{t_1}{1+2\alpha}$ for simplicity. We define $\{A_l^\star\}_{l=1}^{\infty}$ this way in order to use the following inequalities:

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

Using these inequalities, we have the following upper bound on the left hand side of (6):

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

Because $\bigcup_{i=l+1}^{\infty} A_i^\star$ is a superset of $(\bigcup_{i=l}^{\infty} A_i) \cap \left(\bigcup_{i=l+1}^{\infty} A_i^\star\right)$, 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 (6):

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

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

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

Using Lemma 3.7, the above inequality is equivalent to

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

Interchanging the order of the summations on both sides gives

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

Using equations $\sum_{l=1}^{i-1} t_l = \frac{t_i - t_1}{2\alpha}$ and $\sum_{l=1}^{i} t_l = \frac{t_{i+1}-t_1}{2\alpha}$, the above inequality is equivalent to

$$\sum_{i=1}^{\infty}\left(\sum_{j\in A_i} w_j \frac{t_{i+1}-t_1}{2} + \sum_{j\in A_i^\star} (\beta-\alpha) w_j \frac{t_i - t_1}{2\alpha}\right) \geq \sum_{i=1}^{\infty}\left(\sum_{j\in A_i} w_j c_j^{ALG_i} + \sum_{j\in A_i} \beta w_j \frac{t_i - t_1}{2\alpha}\right).$$

Note that $\sum_{i=1}^{\infty}\sum_{j\in A_i} w_j = \sum_{i\in I} w_j = \sum_{i=1}^{\infty}\sum_{j\in A_i^\star} w_j$, so the total coefficient of t_1 on the left hand side can be expressed as $\sum_{i\in I} w_j\left(\frac{-1}{2} + (\beta-\alpha)\frac{-1}{2\alpha}\right) = \sum_{i\in I} w_j\left(\frac{-\beta}{2\alpha}\right)$, which is exactly the coefficient of t_1 on the right hand side. Therefore, the terms related to t_1 are cancelled. Using $t_{i+1} = (1+2\alpha)t_i$, the above inequality is equivalent to

$$\sum_{i=1}^{\infty}\left(\sum_{j\in A_i} w_j \frac{(1+2\alpha)t_i}{2} + \sum_{j\in A_i^\star} (\beta-\alpha) w_j \frac{t_i}{2\alpha}\right) \geq \sum_{i=1}^{\infty}\left(\sum_{j\in A_i} w_j c_j^{ALG_i} + \sum_{j\in A_i} \beta w_j \frac{t_i}{2\alpha}\right),$$

or equivalently,

$$\sum_{i=1}^{\infty}\sum_{j\in A_i^\star} (\beta-\alpha) w_j \frac{t_i}{2\alpha} \geq \sum_{i=1}^{\infty}\sum_{j\in A_i} w_j c_j^{ALG_i} + \sum_{i=1}^{\infty}\sum_{j\in A_i} w_j t_i \left(\frac{\beta}{2\alpha} - \frac{1+2\alpha}{2}\right).$$

Multiplying both sides by $\frac{2\alpha^2}{(1+2\alpha)(\beta-\alpha)}$ and using the definition of $C(\alpha)(I)$ and $T(0)(I)$, the above inequality is equivalent to the following inequality:

$$\frac{\alpha}{1+2\alpha}\sum_{i=1}^{\infty}\sum_{j\in A_i^\star} w_j t_i \geq \frac{2\alpha^2}{(1+2\alpha)(\beta-\alpha)} C(\alpha)(I) + \frac{\alpha(\beta-\alpha(1+2\alpha))}{(1+2\alpha)(\beta-\alpha)} T(0)(I). \tag{8}$$

By (6), for all $j \in A_i^\star$, $\frac{\alpha t_{i-1}}{1+2\alpha} = \alpha t_{i-1} < c_i^{OPT}$, so the left hand side of (8) is at most $OPT(I)$, and thus (5) holds. □

Similarly, comparing the costs given by OPT and ALG_l for each integer l by using Lemma 3.9 with $S = A_l \cup A_{l+1}$, we obtain the following lemma:

Lemma 3.11.

$$OPT \geq \frac{\alpha}{2\beta} C(\alpha) + \frac{\alpha}{2+4\alpha} T(0). \tag{9}$$

Proof. Using Lemma 3.9 with $S = A_l \cup A_{l+1}$, we obtain

$$\sum_{j \in A_l \cup A_{l+1}} w_j f_{\alpha,\beta}\left(c_j^{OPT}, t_l\right) \geq \sum_{j \in A_l \cup A_{l+1}} w_j f_{\alpha,\beta}\left(c_j^{ALG_l}, t_l\right) = \sum_{j \in A_l} w_j c_j^{ALG_l} + \sum_{j \in A_{l+1}} w_j \beta t_l. \tag{10}$$

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

$$\frac{\beta}{\alpha} x \geq f_{\alpha,\beta}(x, y). \tag{11}$$

Combining (10) and (11), we obtain

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

Summing over all positive integers l,

$$\frac{\beta}{\alpha} \sum_{l=1}^{\infty} \sum_{j \in A_l \cup A_{l+1}} w_j c_j^{OPT} \geq \sum_{l=1}^{\infty} \sum_{j \in A_l} w_j c_j^{ALG_l} + \sum_{l=1}^{\infty} \sum_{j \in A_{l+1}} w_j \beta t_l.$$

Using Lemma 3.7,

$$\frac{\beta}{\alpha} \sum_{l=1}^{\infty} \left(\sum_{j \in A_l} w_j c_j^{OPT} + \sum_{j \in A_{l+1}} w_j c_j^{OPT} \right) \geq \sum_{l=1}^{\infty} \sum_{j \in A_l} w_j c_j^{ALG_l} + \sum_{l=1}^{\infty} \sum_{j \in A_{l+1}} w_j \beta t_l. \tag{12}$$

Note that

$$\sum_{l=1}^{\infty} \sum_{j \in A_l} w_j c_j^{OPT} = OPT, \quad \sum_{l=1}^{\infty} \sum_{j \in A_{l+1}} w_j c_j^{OPT} = OPT - \sum_{j \in A_1} w_j c_j^{OPT},$$

$$\sum_{l=1}^{\infty} \sum_{j \in A_l} w_j c_j^{ALG_l} = C(\alpha), \text{ and}$$

$$\sum_{l=1}^{\infty} \sum_{j \in A_{l+1}} w_j t_l = \sum_{l=2}^{\infty} \sum_{j \in A_l} w_j t_{l-1} = \sum_{l=1}^{\infty} \sum_{j \in A_l} w_j t_{l-1} - \sum_{j \in A_1} w_j t_0 =$$

$$\sum_{l=1}^{\infty} \sum_{j \in A_l} w_j \frac{t_l}{1+2\alpha} - \sum_{j \in A_1} \frac{w_j t_1}{2\alpha+1} = \frac{T(0)}{2\alpha+1} - \sum_{j \in A_1} \frac{w_j t_1}{2\alpha+1},$$

where t_0 is defined to be $\frac{t_1}{1+2\alpha}$ for notational convenience. Therefore, Inequality (12) is equivalent to

$$\frac{2\beta}{\alpha}OPT - \frac{\beta}{\alpha}\sum_{j\in A_1} w_j c_j^{OPT} \geq C(\alpha) + \frac{\beta}{1+2\alpha}T(0) - \beta\sum_{j\in A_1}\frac{w_j t_1}{2\alpha+1}.$$

Condition 2 of t_1 described in Section 3.1.1 gives for all $j \in A_1$, $c_j^{OPT} \geq \frac{t_1}{2\alpha+1}$. Therefore,

$$\frac{\beta}{\alpha}\sum_{j\in A_1} w_j c_j^{OPT} \geq \beta\sum_{j\in A_1} w_j c_j^{OPT} \geq \beta\sum_{j\in A_1}\frac{w_j t_1}{2\alpha+1}.$$

Summing the two inequalities above and multiplying all terms by $\frac{\alpha}{2\beta}$, we obtain the lemma. □

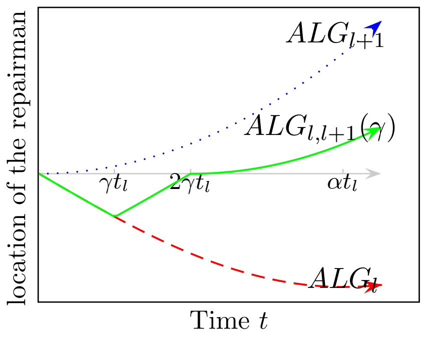
In the previous two lemmas, for each positive integer l, we compare ALG_l with OPT by using Lemma 3.9 to obtain inequalities between $C(\alpha)$, $T(0)$, and OPT. In order to obtain more inequalities, we compare ALG_l with a family of uncountably many algorithms. Since comparing ALG_l with an algorithm gives an inequality between $\{C(r)\}_{r\in[0,\alpha]}$ and $\{T(r)\}_{r\in[0,\alpha]}$, the following lemma gives uncountably many inequalities (of which a finite number will be used to obtain (2d)):

Lemma 3.12. *For any* $\gamma \in \left[0, \frac{\alpha}{2}\right]$,

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

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

Figure 2: Illustration of $ALG_{l,l+1}(\gamma)$



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 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}.

Comparing $ALG_{l,l+1}(\gamma)$ with ALG_l in Lemma 3.9 with $S = A_l \cup A_{l+1}$, we obtain the following inequality:

$$\sum_{j\in A_l\cup A_{l+1}} w_j f_{\alpha,\beta}\left(c_j^{ALG_l}, t_l\right) \le \sum_{j\in A_l\cup A_{l+1}} w_j f_{\alpha,\beta}\left(c_j^{ALG_{l,l+1}(\gamma)}, t_l\right). \tag{13}$$

The left hand side of (13) equals

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

The part of the summation over A_l on the right hand side of (13) is at most

$$\sum_{j\in A_l, c_j^{ALG_l}\le \gamma t_l} w_j c_j^{ALG_l} + \beta \sum_{j\in A_l, c_j^{ALG_l}>\gamma t_l} w_j t_l.$$

The part of the summation over A_{l+1} on the right hand side of Inequality (13) is at most

$$\frac{2\gamma}{1+2\alpha}\sum_{j\in A_{l+1}} w_j t_{l+1} + \sum_{j\in A_{l+1}, c_j^{ALG_{l+1}}\le \frac{\alpha-2\gamma}{1+2\alpha}t_{l+1}} w_j c_j^{ALG_{l+1}} + \frac{\beta-2\gamma}{1+2\alpha}\sum_{j\in A_{l+1}, c_j^{ALG_{l+1}}> \frac{\alpha-2\gamma}{1+2\alpha}t_{l+1}} w_j t_{l+1}$$

where we have used the equation $t_{l+1} = (1+2\alpha)t_l$. This expression is not tight only if the server under $ALG_{l,l+1}(\gamma)$ happens to visit some locations of requests j in A_{l+1} earlier than $2\gamma t_l + c_j^{ALG_{l+1}}$. Summing over all positive integers l and using Lemma 3.7, we obtain the lemma. □

Now we are ready to prove Theorem 3.1, which we repeat here:

Theorem 3.1. Let (α, β) be a pair of real numbers satisfying $\alpha \in (0,1]$ and $\beta \ge \alpha$. For any positive integer N, the objective value of $\mathrm{LP}^{det}_{\alpha,\beta}(N)$ is an upper bound on the competitive ratio of $PAC_{\alpha,\beta}$.

Proof. The idea is to use $\left\{\frac{C(r)}{OPT}\right\}_{r\in[0,\alpha]}$ and $\left\{\frac{T(r)}{OPT}\right\}_{r\in[0,\alpha]}$ as variables in a linear program. Since there are uncountably infinite many variables in $\left\{\frac{C(r)}{OPT}\right\}_{r\in[0,\alpha]}$ and in $\left\{\frac{T(r)}{OPT}\right\}_{r\in[0,\alpha]}$, we divide $[0,\alpha]$ into $N+1$ arithmetic steps, and for all $i = 0, 1, \ldots, N$, we use C_i to represent $C(\frac{i\alpha}{N})/OPT$ and T_i to represent $T(\frac{i\alpha}{N})/OPT$. By definition, $C_0 = T_N = 0$, so there is only a total of $2N$ non-negative real variables $\{C_i\}_{i=1}^N$ and $\{T_i\}_{i=0}^{N-1}$.

Due to Inequality (3), $C_N + T_0$, the objective of $\mathrm{LP}^{det}_{\alpha,\beta}(N)$, is an upper bound on the competitive ratio of $PAC_{\alpha,\beta}$. Therefore, what is left is to show that the linear constraints are all valid for all problem instances I. Lemma 3.8 proves Constraints (2a) and (2b). Lemma 3.10 proves Constraint (2e). Lemma 3.11 proves Constraint (2c). For each $i = 0, 1, \ldots, \lfloor\frac{N}{2}\rfloor$, Lemma 3.12 with $\gamma = \frac{i\alpha}{N}$ gives

$$C(\alpha) + \frac{\beta - 2\frac{i\alpha}{N}}{1+2\alpha}T(0) \le C\left(\frac{i\alpha}{N}\right) + \beta T\left(\frac{i\alpha}{N}\right) + C\left(\frac{\alpha - 2\frac{i\alpha}{N}}{1+2\alpha}\right) + \frac{\beta - 2\frac{i\alpha}{N}}{1+2\alpha}T\left(\frac{\alpha - 2\frac{i\alpha}{N}}{1+2\alpha}\right).$$

Because $C(r)$ is non-decreasing in r and $T(r)$ is non-increasing in r, the right hand side of above inequality is less than or equal to

$$C\left(\frac{i\alpha}{N}\right) + \beta T\left(\frac{i\alpha}{N}\right) + C\left(\left\lceil\frac{N-2i}{1+2\alpha}\right\rceil\frac{\alpha}{N}\right) + \frac{\beta - 2\frac{i\alpha}{N}}{1+2\alpha}T\left(\left\lfloor\frac{N-2i}{1+2\alpha}\right\rfloor\frac{\alpha}{N}\right),$$

which proves Constraints (2d). This concludes the proof. $\square$

In the proof of Theorem 3.1, we first find inequalities related to terms of the forms

$$\sum_{j\in A_l, c_j^{ALG_l}\le rt_l} w_j c_j^{ALG_l} \qquad \text{and} \qquad \sum_{j\in A_l, c_j^{ALG_l}>rt_l} w_j t_l$$

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. Indeed, denote by $l_{\max}$ 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 optimal objective value of $\mathrm{LP}^{det}_{\alpha,\beta}(N)$ with $\alpha = \beta = 1$ when $l_{\max}$ increases. Therefore, one cannot hope to find significantly better upper bounds by considering each of the linear inequalities separately. Now we prove Proposition 3.4, which we repeat here:

Proposition 3.4. For any $\alpha \in (0,1]$, when $\beta \geq 2\alpha^2 + 3\alpha$, the competitive ratio of $PAC_{\alpha,\beta}$ is at most $\frac{(1+2\alpha)(1+\alpha)}{\alpha}$.

Proof. Recall that (5) states

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

Considering Lemma 3.8 with $r = 0$ and $r' = \alpha$, we obtain

$$0 \geq C(\alpha) - \alpha T(0). \tag{14}$$

The inequality

$$\frac{(1+2\alpha)(1+\alpha)}{\alpha} \times (5) + \frac{\beta - (2\alpha^2+3\alpha)}{\beta-\alpha} \times (14)$$

gives

$$\begin{aligned}\frac{(1+2\alpha)(1+\alpha)}{\alpha}OPT \geq &\frac{2\alpha(1+\alpha)}{\beta-\alpha}C(\alpha) + \frac{(\beta-\alpha(1+2\alpha))(1+\alpha)}{\beta-\alpha}T(0)\\ &+ \frac{\beta-(2\alpha^2+3\alpha)}{\beta-\alpha}C(\alpha) - \frac{(\beta-(2\alpha^2+3\alpha))\alpha}{\beta-\alpha}T(0),\end{aligned}$$

where for the right hand side of the inequality, we can put the two terms related to $C(\alpha)$ together and the two terms related to $T(0)$ together and obtain a simplified expression of $C(\alpha) + T(0)$. This inequality, together with Inequality (3), proves the proposition. $\square$

3.1.2.2 Lower Bounds on the Competitive Ratios

In this section, we prove Theorem 3.5. For the case of general β and the case of $\beta \to \infty$, we need to introduce Lemmas 3.13 and 3.14, respectively.

Lemma 3.13. *For $\alpha \in (0, 1]$, for each $\epsilon > 0$, define $I(\epsilon)$ to be the problem instance that contains only one request with $r_1 = 1$, $l_1 = (1 + 2\alpha)^k \alpha + \epsilon$, and $w_1 = 1$, where k is the smallest integer such that $(1 + 2\alpha)^k \alpha > 1$. Then for all $\beta \geq \alpha$,*

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

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 picked k such that $l_1 > (1 + 2\alpha)^k \alpha > 1 = r_1$. As a result,

$$OPT(I(\epsilon)) \leq l_1 = (1 + 2\alpha)^k \alpha + \epsilon. \tag{15}$$

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

$$PAC_{\alpha,\beta}(I(\epsilon)) = c_1^{PAC_{\alpha,\beta}} \geq t_{k+2} + l_1 = (1 + 2\alpha)^{k+1} + (1 + 2\alpha)^k \alpha + \epsilon > (1 + 2\alpha)^k (1 + 3\alpha). \tag{16}$$

Considering (15) and (16), the lemma holds. □

Lemma 3.14. *For $\alpha \in (0, 1]$, for each $\epsilon > 0$, define the problem instance $I(\epsilon)$ to have the following three requests:*

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

where k is the smallest positive integer such that $(1 + 2\alpha)^k \alpha > 1$. Then

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

Proof. The cost of the optimal algorithm is lower than the cost of 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 for $\epsilon < 1$, the completion time of any other request is bounded above by a constant with respect to ϵ, e.g., $3\alpha(1 + 2\alpha)^k \alpha + 1$. In addition, the total weight for the first two requests approaches 0 as $\epsilon \to 0^+$. As a result, as $\epsilon \to 0^+$,

$$OPT(I(\epsilon)) \leq ALG(I(\epsilon)) = \sum_{j=1}^{2} w_j c_j^{ALG} + w_3 c_3^{ALG} \to (1 + 2\alpha)^k \alpha. \tag{17}$$

Now let us consider the online algorithm $PAC_{\alpha,\beta}$. The key observation is that requests 1 and 3 are completed in the last iteration (which is the $(k+2)^{\text{th}}$ 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 $t_1 = 1$, $t_k = (1+2\alpha)^{k-1}$, $t_{k+1} = (1+2\alpha)^k$, and $t_{k+2} = (1+2\alpha)^{k+1}$. For a fixed α and a fixed problem instance $I(\epsilon)$, when β is large enough, $PAC_{\alpha,\beta}$ maximizes the total weight of requests that can be completed at each iteration. Because of the definition of k, $(1+2\alpha)^k\alpha^2 > \alpha$. Therefore, for all small enough ϵ, we have $|l_1| > \alpha$. For such ϵ, the first request cannot be completed in the first k iterations because $t_k\alpha = (1+2\alpha)^{k-1}\alpha^2 \leq \alpha < |l_1|$, where the first inequality comes from the fact that k is the "smallest" positive integer such that $(1+2\alpha)^k\alpha > 1$. Therefore, the first request is in R_{k+1}. At time t_{k+1}, the auxiliary offline algorithm ALG_{k+1} will complete only the second request, because its weight ϵ is greater than that of the first request ϵ^2, and it is impossible to complete both of the first two requests. As a result, the first request will not be completed in the second iteration, and thus is in R_{k+2}. Therefore, the set R_{k+2} contains the first and the third requests.

At time t_{k+2}, the only way to complete both requests in R_{k+2} (the first and the third requests in $I(\epsilon)$) is to visit l_1 first. Thus,

$$\lim_{\beta\to\infty} c_{k+2}^{PAC_{\alpha,\beta}} \geq t_{k+2} + 2|l_1| + |l_3| = (1+2\alpha)^{k+1}(1+\alpha) - \epsilon.$$

Therefore,

$$\lim_{\beta\to\infty} PAC_{\alpha,\beta}(I(\epsilon)) \geq \lim_{\beta\to\infty} w_3 c_3^{PAC_{\alpha,\beta}} = (1+2\alpha)^{k+1}(1+\alpha) - \epsilon. \tag{18}$$

Combining (17) and (18), the lemma holds. □

Now we are ready to prove Theorem 3.5, which we repeat here:

Theorem 3.5. 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 $\frac{(1+\alpha)(1+2\alpha)}{\alpha}$.

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

Lemma 3.13 proves the first statement of the theorem.

Lemma 3.14 implies the following statement: For any real number $\rho < \frac{(1+\alpha)(1+2\alpha)}{\alpha}$, there exists an adversarial problem instance I on which the ratio between $PAC_{\alpha,\beta}(I)$ and $OPT(I)$ is greater than ρ for all large enough β. Thus, Lemma 3.14 proves the second statement of the theorem, which completes the proof. □

3.2 Randomized (α,β)-Plan-and-Commit

We now turn our attention to randomized online algorithms. In Section 3.2.1, we look at the family of randomized online algorithms, parameterized by a pair of numbers (α,β) satisfying $\alpha \in (0,1]$ and $\beta \in [\alpha,\infty)$. In Section 3.2.2, we analyze the proposed algorithms and provide upper and lower bounds on the competitive ratios, and determine the parameters (α,β) that lead to the smallest provable competitive ratios.

3.2.1 The Algorithms

For each pair of real numbers (α, β) such that $\alpha \in (0, 1]$ and $\beta \in [\alpha, \infty)$ we define in Algorithm 2 a randomized online algorithm called $RPAC_{\alpha,\beta}$.

Algorithm 2 Randomized (α, β)-Plan-and-Commit Algorithm ($RPAC_{\alpha,\beta}$) for the online WTRP

1. Initialization
   (a) At time $t = 0$, set $\tau \leftarrow \min\{d(D, l_j) | j \in I_0, l_j \neq D\}$. By convention, if $\{j \in I_0, l_j \neq D\}$ is an empty set, then set $\tau \leftarrow \infty$.
   (b) At time $t \in (0, \tau)$, if a request is revealed, then set $t_1 \leftarrow t$ and end the initialization phase.
   (c) At time $t = \tau$, set $t_1 \leftarrow \tau$ and end the initialization phase.

   At the time when the initialization phase ends: For all positive integer l, set $t_l \leftarrow t_1 \times (1 + 2\alpha)^{l-1}$. Draw $\omega \sim u[0, 1)$ and for all positive integer l, set $t_l \leftarrow t_l \times (1 + 2\alpha)^{\omega}$. Wait until time t_1, define $R_1 \triangleq I_{t_1}$.
2. Repeat for $l = 1, 2, \ldots$, do what $PAC_{\alpha,\beta}$ does with the t_l computed by this algorithm.

$RPAC_{\alpha,\beta}$ is quite similar to the deterministic online $PAC_{\alpha,\beta}$. The only difference is that, at the end of the initialization phase, after getting the geometric time series $\{t_l\}_{l=1}^{\infty}$ similar to what the deterministic algorithm $PAC_{\alpha,\beta}$ does, $RPAC_{\alpha,\beta}$ draws a random variable ω uniformly from $[0, 1)$, and multiplies each t_l by the same random variable $(1 + 2\alpha)^{\omega}$ for all l, i.e., setting $t_l \leftarrow t_l \times (1 + 2\alpha)^{\omega}$. Therefore, for the $RPAC_{\alpha,\beta}$, t_1 is a random variable. Clearly, Conditions 1 and 2 about t_1 described in Section 3.1.1 are satisfied for all realization of ω. Furthermore, for all realization of ω, (1) is satisfied.

3.2.2 Competitive Analysis

Again we first describe our results and provide the proofs regarding the upper bounds and lower bounds in Subsections 3.2.2.1 and 3.2.2.2, respectively.

Upper Bounds: Similar to the results for the deterministic online algorithms, for general pairs of (α, β) satisfying $\alpha \in (0, 1]$ and $\beta \geq \alpha$, the lowest upper bounds that we prove are related to the following linear program, which is parameterized by a positive integer N:

For $\beta = \alpha$

$$\underset{(C_1,C_2,\ldots C_N,T_0,T_1,\ldots,T_{N-1})\in\mathbb{R}^{2N}_{\geq 0}}{\text{maximize}} \quad C_N + T_0 \qquad (\text{LP}^{rand}_{\alpha,\beta}(N))$$

subject to

$$\text{Constraints (2a), (2b), (2c), (2d)}$$

where C_0 and T_N are defined to be 0.

For $\beta > \alpha$

$$\underset{(C_1,C_2,\ldots C_N,T_0,T_1,\ldots,T_{N-1})\in\mathbb{R}^{2N}_{\geq 0}}{\text{maximize}} \quad C_N + T_0 \qquad (\text{LP}^{rand}_{\alpha,\beta}(N))$$

subject to

$$\text{Constraints (2a), (2b), (2c), (2d)}$$
$$1 \geq \frac{\alpha\ln(1+2\alpha)}{\beta-\alpha}C_N + \frac{(\beta-\alpha(1+2\alpha))\ln(1+2\alpha)}{2(\beta-\alpha)}T_0 \tag{19}$$

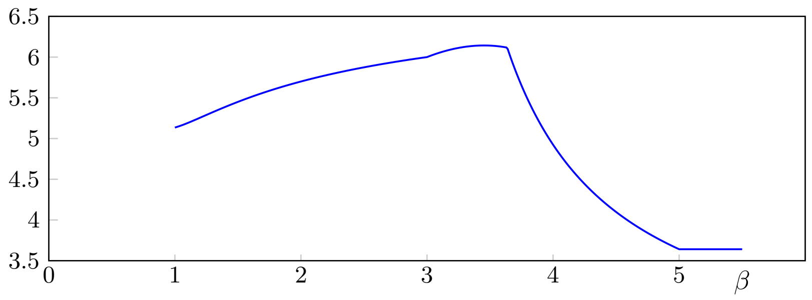
where C_0 and T_N are defined to be 0.

The main result can then be stated as:

Theorem 3.15. *Let (α,β) be a pair of real numbers satisfying $\alpha \in (0,1]$ and $\beta \geq \alpha$. For any positive integer N, the objective value of $LP^{rand}_{\alpha,\beta}(N)$ is an upper bound on the competitive ratio of $RPAC_{\alpha,\beta}$.*

Similar to the case of the deterministic algorithms, Theorem 3.15 allows us to select parameters α and β such that the upper bound on the competitive ratio is minimized. We numerically calculate the optimal objective value for all pairs of α, β when they are both multiples of 0.01 by using Gurobi. For example, Figure 3 illustrates the optimal objective values of $\text{LP}^{rand}_{\alpha,\beta}(N)$ with $N = 10000$ and $\alpha = 1$ for different values of β.

Figure 3: Optimal objective values of $\text{LP}^{rand}_{\alpha,\beta}(N)$ with $N = 10000$ and $\alpha = 1$ for different values of β



Again the observation of all our 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 believe that for all $\beta \geq 5$, $(1,\beta)$ minimizes the provable competitive ratio in our analysis. In fact, for the case $\alpha = 1$ and $\beta \geq 5$, we have a closed-form expression for the competitive ratio. More generally, for any $\alpha \in (0,1]$, when $\beta \geq 2\alpha^2 + 3\alpha$, our best provable upper bound on the competitive ratio has a closed-form expression, as formalized in the following proposition:

Proposition 3.16. *For any $\alpha \in (0,1]$, when $\beta \geq 2\alpha^2 + 3\alpha$, the competitive ratio of $RPAC_{\alpha,\beta}$ is at most $\frac{2(1+\alpha)}{\ln(1+2\alpha)}$.*

The expression of the upper bound on the competitive ratio given by Proposition 3.16 is the same as that of the existing online algorithms INTERVAL [31] and BREAK [23], 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 ($(1, 1+\sqrt{2}]$). 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 existing 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.

In summary, the following corollary describes the combinations of (α, β) that achieve the smallest competitive ratio that we can obtain:

Corollary 3.17. *For any $\beta \geq 5$, the competitive ratio of $RPAC_{1,\beta}$ is at most $4/\ln(3) \approx 3.64$.*

Proof. Proposition 3.16 with $\alpha = 1$ proves the corollary. □

Lower Bounds: Regarding lower bounds, we have the following theorem:

Theorem 3.18. *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 second statement of Theorem 3.18 together with Proposition 3.16 show that the analysis is tight when $\beta \to \infty$. Furthermore, for the algorithm that achieves the best provable upper bound on the competitive ratio, $PAC_{1,\beta}$ with $\beta \geq 5$, we have the following corollary for its lower bound:

Corollary 3.19. *For all $\beta \geq 5$, the competitive ratio of $PAC_{1,\beta}$ is at least $1+\frac{2}{\ln 3} > 2.82$.*

Proof. Theorem 3.18 with $\alpha = 1$ proves this corollary. □

3.2.2.1 Upper Bounds on the Competitive Ratios

The proofs related to the upper bounds are similar to those in Section 3.1.2.1. Here we highlight the differences. First, similar to Section 3.1.2.1, we assume, without loss of generality, that no request j has both $r_j = 0$ and $l_j = D$. In what follows, we first prove Theorem 3.15 and then Proposition 3.16.

For proving Theorem 3.15, we must first note that for any realization of ω, Conditions 1 and 2 about t_1 described in Section 3.1.1 are satisfied. Therefore, for all ω, Lemma 3.7 is valid for $RPAC_{\alpha,\beta}$.

Due to Lemma 3.7 and (1), for any problem instance I, $RPAC_{\alpha,\beta}(I)$ has the following upper bound:

$$RPAC_{\alpha,\beta}(I) \leq \sum_{l=1}^{\infty} \sum_{j \in A_l} w_j c_j^{ALG_l} + \sum_{l=1}^{\infty} \sum_{j \in A_l} w_j t_l = C(\alpha)(I) + T(0)(I), \qquad (20)$$

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

$$C(r)(I) \triangleq \sum_{l=1}^{\infty} \mathbb{E}\left[\sum_{j \in A_l, c_j^{ALG_l} \leq r t_l} w_j c_j^{ALG_l}\right] \quad \text{and} \quad T(r)(I) \triangleq \sum_{l=1}^{\infty} \mathbb{E}\left[\sum_{j \in A_l, c_j^{ALG_l} > r t_l} w_j t_l\right].$$

Unlike the deterministic algorithm $PAC_{\alpha,\beta}(I)$, here the notations $\{C(r)(I)\}_{r\in[0,\alpha]}$ and $\{T(r)(I)\}_{r\in[0,\alpha]}$ represent the expected values of the corresponding functions. Note that due to (20), $\sup_I \frac{C(\alpha)(I)}{OPT(I)} + \frac{T(0)(I)}{OPT(I)}$ is an upper bound on the competitive ratio. In order to find a small upper bound on $\sup_I \frac{C(\alpha)(I)}{OPT(I)} + \frac{T(0)(I)}{OPT(I)}$, we find inequalities between $\{T(r)(I)\}_{r\in[0,\alpha]}$, $\{C(r)(I)\}_{r\in[0,\alpha]}$ and $OPT(I)$ that are valid for all problem instances I. For notational convenience, we drop the parameter I when it is clear from context. Clearly, the inequalities described in Lemmas 3.8-3.12 are still valid when conditioned on any particular realization of ω. Taking the expectations, those inequalities are all valid for the notations $\{T(r)(I)\}_{r\in[0,\alpha]}$, $\{C(r)(I)\}_{r\in[0,\alpha]}$ defined here. Here we slightly abuse notations and refer to Lemmas 3.8-3.12 as with the notations $\{T(r)(I)\}_{r\in[0,\alpha]}$, $\{C(r)(I)\}_{r\in[0,\alpha]}$ defined here for $RPAC_{\alpha,\beta}$.

To find better upper bounds, we improve upon Lemma 3.10 and prove the following lemma:

Lemma 3.20. *When $\beta > \alpha$,*

$$OPT \geq \frac{\alpha \ln(1+2\alpha)}{\beta - \alpha} C(\alpha) + \frac{(\beta - \alpha(1+2\alpha))\ln(1+2\alpha)}{2(\beta-\alpha)} T(0). \tag{21}$$

Proof. We follow the proof of Lemma 3.10. For $RPAC_{\alpha,\beta}$, since (8) is valid for any realization of ω, it is also valid when we take expectations on both sides. For each $j \in I$, the minimum value αt_l such that $c_j^{OPT} \leq \alpha t_l$ has the following expected value:

$$\int_0^1 (1+2\alpha)^x c_j^{OPT} \, \mathrm{d}x = \frac{2\alpha}{\ln(1+2\alpha)} c_j^{OPT}. \tag{22}$$

This equation gives us the following equation:

$$\mathbb{E}\left(\sum_{i=1}^{\infty} \sum_{j \in A_i^\star} w_j t_i\right) = \sum_{j \in I} \frac{2}{\ln(1+2\alpha)} w_j c_j^{OPT} = \frac{2}{\ln(1+2\alpha)} OPT(I). \tag{23}$$

Taking expectation on both sides of (8), using (23), and rearranging terms properly, we obtain (21), which proves the lemma. □

Now we are ready to prove Theorem 3.15, which we repeat here:

Theorem 3.15. Let (α, β) be a pair of real numbers satisfying $\alpha \in (0, 1]$ and $\beta \geq \alpha$. For any positive integer N, the objective value of $\mathrm{LP}_{\alpha,\beta}^{rand}(N)$ is an upper bound on the competitive ratio of $RPAC_{\alpha,\beta}$.

Proof. The idea is to use $\left\{\frac{C(r)}{OPT}\right\}_{r\in[0,\alpha]}$ and $\left\{\frac{T(r)}{OPT}\right\}_{r\in[0,\alpha]}$ as variables in a linear program. Since there are uncountably infinite many variables in $\left\{\frac{C(r)}{OPT}\right\}_{r\in[0,\alpha]}$ and in $\left\{\frac{T(r)}{OPT}\right\}_{r\in[0,\alpha]}$, we divide $[0,\alpha]$ into $N+1$ arithmetic steps, and for all $i=0,1,\ldots,N$, we use C_i to represent $C(\frac{i\alpha}{N})/OPT$ and T_i to represent $T(\frac{i\alpha}{N})/OPT$. By definition, $C_0 = T_N = 0$, so there is only a total of $2N$ non-negative real variables $\{C_i\}_{i=1}^{N}$ and $\{T_i\}_{i=0}^{N-1}$.

Due to Inequality (20), $C_N + T_0$, the objective of $\mathrm{LP}_{\alpha,\beta}^{rand}(N)$, is an upper bound on the competitive ratio of $RPAC_{\alpha,\beta}$. Therefore, what is left is to show that the linear constraints are all valid for all problem instances I. Lemma 3.8 proves Constraints (2a) and (2b). Lemma 3.11 proves Constraint (2c). Lemma 3.12 proves the Constraints (2d) for all i. Finally, Lemma 3.20 proves Constraint (19). This concludes the proof. □

Now we prove Proposition 3.16, which we repeat here:

Proposition 3.16. For any $\alpha \in (0,1]$, when $\beta \geq 2\alpha^2 + 3\alpha$, the competitive ratio of $RPAC_{\alpha,\beta}$ is at most $\frac{2(1+\alpha)}{\ln(1+2\alpha)}$.

Proof. Considering Lemma 3.8 with $r=0$ and $r'=\alpha$, we obtain

$$0 \geq C(\alpha) - \alpha T(0). \tag{24}$$

The inequality

$$\frac{2(1+\alpha)}{\ln(1+2\alpha)} \times (21) + \frac{\beta - (2\alpha^2+3\alpha)}{\beta - \alpha} \times (24)$$

and Inequality (20) prove the proposition. □

3.2.2.2 Lower Bounds on the Competitive Ratios

In this section, we prove Theorem 3.18. For the case of general β and the case of $\beta \to \infty$, we need to introduce Lemmas 3.21 and 3.22, respectively.

Lemma 3.21. *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, $OPT(I) = 1$. On the other hand, the expected cost of $RPAC_{\alpha,\beta}$ is

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

□

Lemma 3.22. *For each $\epsilon > 0$, define $I(\epsilon)$ to be the problem instance with the following $2M+1$ requests:*

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

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 cost of 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 at most 3. Moreover, as $\epsilon \to 0^+$, the total weight of the first $2M$ requests approaches 0. As a result, as $\epsilon \to 0^+$, the cost of OPT has the following upper bound:

$$OPT(I(\epsilon)) \leq ALG(I(\epsilon)) \to w_{2M+1} c_{2M+1}^{ALG} = \alpha. \tag{25}$$

Let us now consider the randomized algorithm $RPAC_{\alpha,\beta}$. 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 $(M+1)^{\text{th}}$ through the $2M^{\text{th}}$ requests 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 be the smallest integer such that $t_k \geq 1$. The expected value of t_k is

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

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

$$c_{2M+1}^{RPAC_{\alpha,\beta}} = t_k + c_{2M+1}^{ALG_k} \geq (1+\alpha)t_k - 2\epsilon^2.$$

Considering 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:

$$RPAC_{\alpha,\beta}(I(\epsilon)) \geq \mathbb{E}\left(w_{2M+1} c_{2M+1}^{RPAC_{\alpha,\beta}}\right) \geq \mathbb{E}\left((1+\alpha)t_k - 2\epsilon^2\right) \to \frac{2\alpha(1+\alpha)}{\ln(1+2\alpha)}. \tag{26}$$

Considering (25) and (26), the lemma holds. □

Now we are ready to prove Theorem 3.18, which we repeat here:

Theorem 3.18. 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 $\frac{2(1+\alpha)}{\ln(1+2\alpha)}$.

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

Lemma 3.21 proves the first statement of the theorem.

Lemma 3.22 implies the following statement: For any real number $\rho < \frac{2(1+\alpha)}{\ln(1+2\alpha)}$, there exists an adversarial problem instance I on which the ratio between $RPAC_{\alpha,\beta}(I)$ and $OPT(I)$ is greater than ρ for all large enough β. Thus, Lemma 3.14 proves the second statement of the theorem, which completes the proof. □

3.3 Probabilistic Version

In this section, we consider a probabilistic version of the online WTRP discussed in Jaillet and Wagner [25] and which makes the following stochastic assumptions:

Assumption 3.23 (Locations)**.** *The metric space is an M-dimensional Euclidean space, and the locations are independently drawn from an identical distribution over a compact support in the metric space.*

Assumption 3.24 (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 non-negative support with a finite mean and variance.*

Assumption 3.25 (Weights)**.** *The weight of each request has positive upper and lower bounds, i.e., there exists $0 < \underline{w} < \bar{w}$ such that for all $j \in [n]$, $w_j \in [\underline{w}, \bar{w}]$.*

We are specifically interested here in the asymptotic behavior of a simple online algorithm, called *ReOpt*, that simply re-optimizes the route of the server whenever a new request is released. The algorithm can be formally described as follows:

Algorithm 3 Online algorithm *ReOpt*

```
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 to minimize
   the total weighted completion time of the released but not completed requests
   (starting at the server's current location).
```

The following theorem is a special case of Theorem 3 in [25]:

Theorem 3.26. *(from [25]) For any sequence of problem instances $\{I^i\}_{i=1}^{\infty}$ where for any $i < n$, I^i is the set of the first i requests in I^n, almost surely,*

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

It turns out that the proof of Theorem 3 as given in [25] is problematic. Indeed, it relies on a lemma (Lemma 5 in [25]) whose full proof needs the assumption that, for all problem instances,

$$\max_{j\in\{1,2,\ldots,n-1\}} c_j^{ReOpt_{n-1}} \leq r_n + \frac{3}{2} TSP_{n-1}. \tag{27}$$

To see that (27) does not necessary hold, consider the problem instance that has the following five requests in 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 four requests are released, 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 four requests has a length of 40. Therefore, $r_n + \frac{3}{2} TSP_{n-1} = 61 < 66$, which contradicts (27). Despite best efforts, we haven't been able to find a way to circumvent that issue within the proof framework given in [25]. Instead we provide here an alternative and complete proof for Theorem 3.26.

In order to do so, let TSP_i be 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 $ReOpt(I^n)$ to TSP_n:

Lemma 3.27.

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

In order to prove Lemma 3.27, let us define $ReOpt_i$, for any positive integer i, as the route of the server under $ReOpt$ when the problem instance consists of only the requests in I^i. The proof uses two intermediate lemmas.

The first one states 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 3.28. *For any integer $k \geq 0$, $1 \leq i \leq n$,*

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

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

Proof. 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 $\frac{1}{2}TSP_i$ and for the third step is TSP_i. Therefore, the completion time is at most $t + \frac{3}{2}TSP_i$ 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

$$\mathrm{Alt}_t(I^i) \le \sum_{c_j^{ReOpt_i} \le t} w_j c_j^{ReOpt_i} + \sum_{c_j^{ReOpt_i} > t} w_j \left(t + \frac{3}{2}TSP_i \right)$$

where we restrict the summations to $j \in [i]$ throughout 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,

$$\begin{aligned}\sum_{c_j^{ReOpt_i} > t} w_j \left(t + \frac{3}{2}TSP_i \right) &\ge \sum_{c_j^{ReOpt_i} > t} w_j c_j^{ReOpt_i} \\ &\ge \sum_{t+3TSP_i \ge c_j^{ReOpt_i} > t} w_j t + \sum_{c_j^{ReOpt_i} > t+3TSP_i} w_j (t + 3TSP_i).\end{aligned}$$

Canceling the same term $w_j t$ and dividing both sides by $3TSP_i$, we obtain

$$\frac{1}{2} \sum_{c_j^{ReOpt_i} > t} w_j \ge \sum_{c_j^{ReOpt_i} > t+3TSP_i} w_j.$$

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

□

The second intermediate lemma gives an iterative relation for $\{ReOpt_i(I^i)\}_{i=1}^{\infty}$:

Lemma 3.29. *For any positive integers i and k,*

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

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

1. Follow $ReOpt_i$ until time $r_i + 3kTSP_i$. (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}.

For all $j \in [i]$, if $c_j^{ReOpt_i} \le \max\{r_i + 3kTSP_i, r_{i+1}\}$, then $c_j^{\mathrm{Alt}} = c_j^{ReOpt_i}$; otherwise, $c_j^{ReOpt_i} > \max\{r_i + 3kTSP_i, r_{i+1}\}$ and thus

$$c_j^{\mathrm{Alt}} \le \max\{r_i + 3kTSP_i, r_{i+1}\} + \frac{1}{2}TSP_i + TSP_{i+1} < c_j^{ReOpt_i} + \frac{3}{2}TSP_{i+1}.$$

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

$$c_{i+1}^{\mathrm{Alt}} \leq \max\{r_i + 3kTSP_i, r_{i+1}\} + \frac{1}{2}TSP_i + TSP_{i+1}$$
$$\leq r_{i+1} + 3kTSP_i + \frac{1}{2}TSP_i + TSP_{i+1} \leq r_{i+1} + \left(3k + \frac{3}{2}\right)TSP_{i+1}.$$

According to Lemma 3.28, the total weight of requests in I^i such that $c_j^{ReOpt_i} > r_i + 3kTSP_i$ is at most $2^{-k}\sum_{j=1}^{i} w_j$. As a result, we have

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

According to the definition of $ReOpt_{i+1}$, we have $ReOpt_{i+1}(I^{i+1}) \leq \mathrm{Alt}(I^{i+1})$, which completes the proof. □

Now we are ready to prove Lemma 3.27, which we repeat here:

Lemma 3.27.

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

Proof. According to Lemma 3.29 and mathematical induction, for any positive integer k,

$$ReOpt(I^n) = ReOpt_n(I^n) \leq \sum_{j=1}^{n} w_j\left(r_j + \left(3k + \frac{3}{2} + \frac{3}{2}n2^{-k}\right)TSP_n\right).$$

Setting $k = \lceil \log n \rceil$, we have

$$ReOpt(I^n) \leq \sum_{j=1}^{n} w_j\left(r_j + \left(3(\log n + 1) + \frac{3}{2} + \frac{3}{2}n2^{-\log n}\right)TSP_n\right)$$
$$= \sum_{j=1}^{n} w_j\left(r_j + (3\log n + 6)\, TSP_n\right). \qquad (n2^{-\log n} = 1)$$

□

We can now complete the proof of Theorem 3.26, which we repeat here:

Theorem 3.26. For any sequence of problem instances $\{I^i\}_{i=1}^{\infty}$ where for any $i < n$, I^i is the set of the first i requests in I^n, almost surely,

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

Proof. For all $j \in [n]$, we have $c_j \geq r_j$. Thus, $OPT(I^n) \geq \sum_{j=1}^{n} w_j r_j$. With this inequality, Lemma 3.27, and Assumption 3.25, we have

$$\frac{ReOpt(I^n)}{OPT(I^n)} \leq 1 + \frac{\sum_{j=1}^{n} w_j O(\log n)\, TSP_n}{\sum_{j=1}^{n} w_j r_j} \leq 1 + \frac{n\bar{w}O(\log n)TSP_n}{\underline{w}\sum_{j=1}^{n} r_j}.$$

According to Beardwood *et al.* [8] and Assumption 3.23, $TSP_n = O(n^{1-\frac{1}{M}})$ almost surely. On the other hand, according to Assumption 3.24, $\sum_{j=1}^{n} r_j$ 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,

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

which approaches 0 as n goes to infinity. $\square$

4 Online Scheduling with Multi-State Machines

In this section, we develop online algorithms for the general online scheduling problems with multi-state machines, as defined in Section 2.1, and based on the best (α, β) pair found in the previous section. We assume $r_1 > 0$ to simplify the initialization phase 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 $r_1 = 0$.[2] As a minor caveat, we impose some mild technical assumptions as discussed in Appendix A.2.

In Section 4.1, we propose a deterministic online algorithm Plan-and-Commit (PAC) and prove that the competitive ratio is at most 5.14. In Section 4.2, for each $\alpha \leq 1$, we propose a randomized online algorithm Randomized α-Plan-and-Commit ($RPAC_\alpha$). 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

4.1.1 The Algorithm

In this section, we describe the algorithm Plan-and-Commit (PAC), which, when applied to the online WTRP, is $PAC_{1,1}$ with minor modifications. It goes as follows:

The PAC algorithm is again composed of two phases (initialization and iterations):

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

The second phase consists of iterations. For each positive integer l, the l^{th} iteration ($[t_l, t_{l+1}]$) 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 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 phase; for $l \geq 2$, R_l is defined at the end of the $(l-1)^{\text{th}}$ iteration. Here we assume that

[2]This can be done by replacing the initialization phase with one that is similar to the one in Algorithm 1: at time 0, find the minimum non-zero completion time of an active project τ, and then end the initialization phase at the minimum between τ and the minimum non-zero release date of a job.

Algorithm 4 The deterministic Plan-and-Commit algorithm (PAC) for general of online scheduling problems with multi-state machines

```
1. Initialization: Wait until the first job is released (at time $r_1$). At time $r_1$, for all
   positive integers $l$, define $t_l \leftarrow r_1 \times (1+2\alpha)^{l-1}$. Define $R_1 \triangleq K(I_{t_1})$.
2. Repeat for $l = 1, 2, \dots,$
    (a) Plan: At time $t = t_l$, calculate what would an offline algorithm, $ALG_l$, have
        done starting at time 0 to minimize $\sum_{k \in R_l} h_k(\min(x_k, t_l))$.
    (b) Commit:
        • At time $t \in [t_l, 2t_l]$, follow a delayed version (delayed by $t_l$) of algorithm
          $ALG_l$ with the following two minor modifications mentioned in the main text.
        • At time $t \in [2t_l, 3t_l]$, control the machine states so that all machines $i$
          are at their original states $O_i$ at time $3t_l$.
    (c) At time $t_{l+1}$, define $R_{l+1} \triangleq (R_l \setminus A_l) \cup (K(I_{t_{l+1}}) \setminus K(I_{t_l}))$ where $A_l \triangleq \{k | k \in
        R_l, x_k^{ALG_l} \le t_l\}$.
```

such an offline algorithm ALG_l exists, which is not necessarily true if the assumptions in Appendix A.2 are not imposed. See Appendix A for a detailed discussion. Note that at the first iteration, staying at the depot is one of the optimal solutions to the auxiliary offline problem. Therefore, in the first iteration we let ALG_1 be this algorithm and hence

$$A_1 = \emptyset. \tag{28}$$

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 controls 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., $A_l \triangleq \{k | k \in R_l, x_k^{ALG_l} \le t_l\}$. Following the definition, for all $k \in A_l$,

$$x_k^{PAC} \le t_l + x_k^{ALG_l}. \tag{29}$$

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., $R_{l+1} \triangleq (R_l \setminus A_l) \cup (K(I_{t_{l+1}}) \setminus K(I_{t_l}))$.

4.1.2 Competitive Analysis

Since the online WTRP is a special case, the lower bounds proved in Section 3.1.2.2 are still valid for this algorithm. Here we discuss our results regarding the upper bounds on the competitive ratios.

Similar to the results for the online WTRP, for general pairs of (α, β) satisfying $\alpha \in (0,1]$ and $\beta \geq \alpha$, the lowest upper bounds that we obtain are related to the following linear program, which is parameterized by a positive integer N:

$$\underset{X_i\geq 0 \text{ for all } i=1,2,\ldots,N \text{ and } T_{i,j}\geq 0 \text{ for all } i=0,\ldots,N-1, j=1,2,\ldots,N}{\text{maximize}} \quad X_N + T_{0,N} \qquad (\text{LP}^{det}(N))$$

subject to

$$T_{i,j+1} + T_{i,j-1} \leq 2T_{i,j} \qquad \text{for } \begin{cases} i = 0,\ldots,N \\ j = 1,\ldots,N-1 \end{cases} \tag{30a}$$

$$X_{i+1} - X_i \geq T_{i,i} - T_{i+1,i} \qquad \text{for } i = 0,\ldots,N-1 \tag{30b}$$

$$X_{i+1} - X_i \leq T_{i,i+1} - T_{i+1,i+1} \qquad \text{for } i = 0,\ldots,N-1 \tag{30c}$$

$$T_{0,\lfloor \frac{N}{3} \rfloor} + X_N \leq 2. \tag{30d}$$

$$X_N + T_{0,\lfloor \frac{N}{3} \rfloor} \leq X_i + T_{i,N} + X_{\lceil \frac{N-2i}{3} \rceil} + T_{0,\lceil \frac{2i}{3} \rceil} + T_{\lfloor \frac{N-2i}{3} \rfloor, \lceil \frac{N-2i}{3} \rceil} \qquad \text{for } i = 0,\ldots,\left\lfloor \frac{N}{2} \right\rfloor \tag{30e}$$

where X_0 is defined to be 0 and for all $i = 0,1,\ldots,N$, $T_{i,0}$ and $T_{N,i}$ are defined to be 0.

Our main result is the following theorem:

Theorem 4.1. *For any positive integer N, the optimal objective value of $LP^{det}(N)$ is an upper bound on the competitive ratio of PAC.*

Proof. The proof of Theorem 4.1 is similar to that of Theorem 3.1 for the online WTRP. In the following proof, we omit arguments that are essentially the same as that of Theorem 3.1.

We first notice that using a similar argument as the proof of Lemma 3.7, 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, $PAC(I)$ has the following upper bound:

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

where

$$X(r)(I) \triangleq \sum_{l=1}^{\infty} \sum_{k \in A_l, x_k^{ALG_l} \leq rt_l} h_k\left(x_k^{ALG_l}\right) \quad \text{and} \quad T(r,v)(I) \triangleq \sum_{l=1}^{\infty} \sum_{k \in A_l, x_k^{ALG_l} > rt_l} h_k\left(vt_l\right).$$

As a result,

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

is an upper bound on the competitive ratio of PAC. We drop the parameter I and view $\{\frac{X(r)}{OPT}\}_{r\in[0,1]}$ and $\{\frac{T(r,v)}{OPT}\}_{r,v\in[0,1]}$ 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 LP$^{det}(N)$ by dividing $[0,1]$ into $N+1$ arithmetic steps, and for all $i=0,1,\dots,N$, letting X_i represent $X(\frac{i}{N})/OPT$ and for all $i,j=0,1,\dots,N$, letting $T_{i,j}$ represent $T(\frac{i}{N},\frac{j}{N})/OPT$. By definition, for all $i=0,1,\dots,N$, $X_0=T_{i,0}=T_{N,i}=0$. Therefore, there is only a total of N^2+N non-negative real variables $\{X_i\}_{i=1}^{N}$ and $\{T_{i,j}\}_{i=0}^{N-1}{}_{j=1}^{N}$.

Constraints (30a) follow from the concavity of functions h_k. Constraints (30b) and (30c) 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 3.8. The property (28) says that A_1 is empty, and this fact is useful for proving the other constraints. Similar to Lemma 3.9, 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

$$\sum_{k\in A_l\cup A_{l+1}} h_k\left(\min\left(x_k,t_l\right)\right). \tag{31}$$

Similar to Lemma 3.11, comparing cost (31) of ALG_l and that of OPT, we obtain the following linear inequality:

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

This inequality gives us (30d). Similar to Lemma 3.12, for all integers l and all $r\in\left[0,\frac{1}{2}\right]$, define algorithm $ALG_{l,l+1}(r)$ to be the algorithm that does the following.

1. At time $0\le t\le rt_l$, follow 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 (31) of ALG_l and that of $ALG_{l,l+1}(r)$, we obtain

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

Using inequalities that come from h_k being concave, $h_k\left(2rt_l+x_k^{ALG_{l+1}}\right)\le h_k\left(2rt_l\right)+h_k\left(x_k^{ALG_{l+1}}\right)$ 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:

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

For all $i=0,\ldots,\lfloor\frac{N}{2}\rfloor$, the above inequality with $r=\frac{i}{N}$, together with the fact that $X(r)$ is non-decreasing in r, and $T(r,v)$ is non-increasing in r and non-decreasing in v, gives

$$\begin{aligned}
&X(1)+T\left(0,\left\lfloor\frac{N}{3}\right\rfloor\frac{1}{N}\right)\leq X(1)+T\left(0,\frac{1}{3}\right)\\
\leq& X\left(\frac{i}{N}\right)+T\left(\frac{i}{N},1\right)+T\left(0,\frac{2\frac{i}{N}}{3}\right)+X\left(\frac{1-2\frac{i}{N}}{3}\right)+T\left(\frac{1-2\frac{i}{N}}{3},\frac{1-2\frac{i}{N}}{3}\right)\\
\leq& X\left(\frac{i}{N}\right)+T\left(\frac{i}{N},1\right)+T\left(0,\left\lceil\frac{2i}{3}\right\rceil\frac{1}{N}\right)+X\left(\left\lceil\frac{N-2i}{3}\right\rceil\frac{1}{N}\right)\\
&+T\left(\left\lfloor\frac{N-2i}{3}\right\rfloor\frac{1}{N},\left\lceil\frac{N-2i}{3}\right\rceil\frac{1}{N}\right),
\end{aligned}$$

which gives Constraints (30e). Thus, the proof is completed. □

Using Theorem 4.1, we obtain the following upper bound on the competitive ratio of PAC:

Corollary 4.2. *The competitive ratio of PAC is at most* 5.14.

Proof. Using Gurobi, the optimal objective value of $\text{LP}^{det}(N)$ with $N=1200$ is smaller than 5.14 (already taking into account numerical errors). Hence, Theorem 4.1 implies this corollary. □

The next corollary (Corollary 4.3) is weaker than Corollary 4.2 but still shows that the competitive ratio of PAC is lower than the best competitive ratio in the literature for the online WTRP (5.83). We include Corollary 4.3 because we can analytically prove this corollary (proved in Appendix D).

Corollary 4.3. *The competitive ratio of PAC is at most* $39/7\approx 5.57$.

4.2 Randomized α-Plan-and-Commit

4.2.1 The Algorithm

Here we describe the randomized online algorithm α-Plan-and-Commit ($RPAC_\alpha$); see Algorithm 5, which, when applied to the online WTRP, is $RPAC_{1,\infty}$ with minor modifications.

The $RPAC_\alpha$ algorithm 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 phases: initialization and iterations.

In the first phase, $RPAC_\alpha(\omega)$ 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.

Algorithm 5 The Randomized α-Plan-and-Commit ($RPAC_\alpha$) for general online scheduling problems with multi-state machines

```
1. Initialization: Draw $\omega \sim u[0,1)$. Wait until the first job is released (at time $r_1$). At time $r_1$, for all positive integer $l$, define $t_l \leftarrow r_1 \times (1+2\alpha)^{l-1+\omega}$. Define $R_1 \triangleq K(I_{t_1})$.
2. Repeat for $l = 1, 2, \ldots,$
    (a) Plan: At time $t = t_l$, calculate what would an offline algorithm, $ALG_l$, have done starting at time 0 to minimize $\sum_{k \in R_l} (h_k(t_{l+1}) - h_k(t_l)) \mathbb{1}(x_k > \alpha t_l)$.
    (b) Commit:
        • At time $t \in [t_l, (1+\alpha)t_l]$, follow a delayed version (delayed by $t_l$) of algorithm $ALG_l$ with the two minor modifications mentioned in the main text.
        • At time $t \in [(1+\alpha)t_l, (1+2\alpha)t_l]$, control the machine states so that all machines $i$ are at their original states $O_i$ at time $(1+2\alpha)t_l$.
    (c) At time $t_{l+1}$, define $R_{l+1} \triangleq (R_l \setminus A_l) \cup (K(I_{t_{l+1}}) \setminus K(I_{t_l}))$ where $A_l \triangleq \{k | k \in R_l, x_k^{ALG_l} \le t_l\}$.
```

The second phase is composed of iterations. The l^{th} iteration ($[t_l, t_{l+1}]$) 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:

$$\sum_{k \in R_l} (h_k(t_{l+1}) - h_k(t_l)) \mathbb{1}(x_k > \alpha t_l), \tag{32}$$

where $\mathbb{1}$ is the indicator function with value 1 if the argument of the function is a true statement and 0 otherwise; and R_l is defined either in the initialization phase (when $l = 1$) or the previous iteration (when $l \ge 2$). The existence of such an algorithm ALG_l is obvious because the cost (32) 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 and modified version of ALG_l, where the modifications are analogues of those for PAC described in Section 4.1. The algorithm $RPAC_\alpha(\omega)$ defines $A_l \triangleq \{k | k \in R_l, x_k^{ALG_l} \le t_l\}$. By doing so, for all $k \in A_l$,

$$x_k^{RPAC_\alpha(\omega)} \le t_l + x_k^{ALG_l}. \tag{33}$$

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 $R_{l+1} \triangleq (R_l \setminus A_l) \cup (K(I_{t_{l+1}}) \setminus K(I_{t_l}))$.[3]

[3]The sets $K(I_{t_{l+1}})$ and $K(I_{t_l})$ are derived from the definitions of $K(I)$, I_t, t_l, and t_{l+1}.

4.2.2 Competitive Analysis

The main result is the following theorem:

Theorem 4.4. *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 3.9.

Lemma 4.5. *For all positive integers l and any realization $\omega \in [0,1)$:*

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

Proof. Obviously, for all $i \leq l-1$ and $k \in A_i$, $x_k^{RPAC_\alpha(\omega)} < (1+\alpha)t_l$. On the other hand, for any project $k \in K(I) \setminus \bigcup_{i=1}^{l-1} A_i$ such that $x_k^{ALG_l} \leq \alpha t_l$, k is in A_l. According to (33), $x_k^{RPAC_\alpha(\omega)} \leq t_l + x_k^{ALG_l} \leq (1+\alpha)t_l$. Therefore,

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

Similar to Lemma 3.9, ALG_l satisfies the following inequality:

$$\begin{aligned}&\sum_{k\in K(I)\setminus\bigcup_{i=1}^{l-1} A_i} \left(h_k\left(t_{l+1}\right) - h_k\left(t_l\right)\right) \mathbb{1}\left(x_k^{ALG_l} > \alpha t_l\right)\\ \leq &\sum_{k\in K(I)\setminus\bigcup_{i=1}^{l-1} A_i} \left(h_k\left(t_{l+1}\right) - h_k\left(t_l\right)\right) \mathbb{1}\left(x_k^{OPT} > \alpha t_l\right). \qquad (34)\end{aligned}$$

Combining the two inequalities above, we obtain the lemma.

□

Now we are ready to prove Theorem 4.4, which we repeat here:

Theorem 4.4. For all $\alpha \in (0,1]$, the competitive ratio of $RPAC_\alpha$ is $\frac{2(1+\alpha)}{\ln(1+2\alpha)}$.

Proof. The lower bound is simple. For the special case of the online WTRP, $RPAC_\alpha$ is reduced to $RPAC_{\alpha,\infty}$. According to Theorem 3.16, the competitive ratio of $RPAC_\alpha$ 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 $(1+\alpha)t_{l+1} \geq x_k^{RPAC_\alpha(\omega)} > (1+\alpha)t_l$, we have the upper bound on the cost incurred by the project: $h_k\left(x_k^{RPAC_\alpha(\omega)}\right) \leq h_k\left((1+\alpha)t_{l+1}\right) \leq (1+\alpha)h_k\left(t_{l+1}\right)$. Taking the summation over all active projects k and rearranging terms, we have the following upper bound on $RPAC_\alpha(\omega)(I)$:

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

where we have defined $t_0 \triangleq t_1/(1+2\alpha)$ for simplicity. Using Lemma 4.5 and exchanging the order of the double summations, we obtain

$$\frac{RPAC_\alpha(\omega)(I)}{1+\alpha} \leq \sum_{k\in K(I)} \sum_{l=0}^{\infty} h_k\left(t_{l+1}\right) \mathbb{1}\left(\alpha t_{l+1} \geq x_k^{OPT} > \alpha t_l\right). \tag{35}$$

Note that different realizations of ω give different sequences of $\{t_l\}_{l=1}^{\infty}$. Therefore, for a fixed k, the integer l such that $\alpha t_{l+1} \geq x_k^{OPT} > \alpha t_l$ can be different. However, due to the distribution of ω, we know that when l is the integer such that $\alpha t_{l+1} \geq x_k^{OPT} > \alpha t_l$, the distribution of αt_{l+1} is the same as that of $x_k^{OPT}(1+2\alpha)^x$ where x is uniform over $[0,1)$. Furthermore, because h_k is concave and $h_k(0)=0$, for $\alpha t_{l+1} \geq x_k^{OPT}$, we have

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

Therefore, taking the expectation on both sides of (35), we obtain

$$\mathbb{E}\left(RPAC_\alpha(\omega)(I)\right) \leq \frac{1+\alpha}{\alpha} \sum_{k\in K(I)} \left(\int_0^1 (1+2\alpha)^x \,\mathrm{d}x\right) h_k\left(x_k^{OPT}\right) = \frac{2(1+\alpha)}{\ln(1+2\alpha)} OPT(I). \tag{36}$$

□

Using Theorem 4.4, we have the following corollary regarding the lowest upper bound on the competitive ratios that we can prove:

Corollary 4.6. *The competitive ratio of $RPAC_1$ is $4/\ln(3) \approx 3.64$.*

Proof. Theorem 4.4 with $\alpha = 1$ proves that the competitive ratio is at most $4/\ln(3)$. Theorem 3.18 with $\alpha = 1$ proves that the competitive ratio is at least $4/\ln(3)$. Thus the corollary holds. □

Remark 4.7. 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:

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

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 3.9 to prove (34), 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 (36), 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 quota-collecting 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 provided a complete proof that *ReOpt* is almost surely asymptotically optimal for the online WTRP.

Even though we have made progress in considering a new 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 are the competitive ratios of $PAC_{\alpha,\beta}$ and $RPAC_{\alpha,\beta}$ for the online WTRP? What is the competitive ratio of PAC for general online scheduling problems with multi-state machines?

Either a tighter analysis of the algorithm or a better adversarial problem instance, or both, could address this open problem.

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

Although *ReOpt* is almost surely asymptotically optimal under some stochastic assumptions, we do not know whether *ReOpt* has a constant competitive ratio under an adversarial model (for both the online WTRP and the general online scheduling problems with multi-state machines).

Open Problem 3. What are 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 are the competitive ratios of the best-possible deterministic and randomized online algorithms for the online WTRP for general metric spaces? What about the competitive ratio for general online scheduling problems with multi-state machines?

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

References

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

[2] A. Allahverdi, C. Ng, T. Cheng, and M.Y. Kovalyov, A survey of scheduling problems with setup times or costs, Eur J Oper Res 187 (2008), 985–1032.

[3] E.J. Anderson and C.N. Potts, Online scheduling of a single machine to minimize total weighted completion time, Math Oper Res 29 (2004), 686–697.

[4] A. Archer and A. Blasiak, Improved approximation algorithms for the minimum latency problem via prize-collecting strolls, Proc 21st Ann ACM-SIAM Symp Discr Algorithms, Society for Industrial and Applied Mathematics, 2010, pp. 429–447.

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

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

[7] G. Ausiello, L. Laura, and E. Pini, 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, 2006.

[8] J. Beardwood, J.H. Halton, and J.M. Hammersley, The shortest path through many points, Math Proc Cambridge Philosophical Soc, Vol. 55, Cambridge Univ Press, 1959, pp. 299–327.

[9] M. Blom, S.O. Krumke, W.E. de Paepe, and L. Stougie, The online TSP against fair adversaries, INFORMS J Comput 13 (2001), 138–148.

[10] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan, The minimum latency problem, Proc 26th Ann ACM Symp Theory Comput, Association for Computing Machinery, 1994, pp. 163–171.

[11] V. Bonifaci, M. Lipmann, and L. Stougie, Online multi-server dial-a-ride problems, TU/e, Eindhoven University of Technology, Department of Mathematics and Computing Science, 2006.

[12] C. Chung, T. Nonner, and A. Souza, SRPT is 1.86-competitive for completion time scheduling, Proc 21st Ann ACM-SIAM Symp Discr Algorithms, Society for Industrial and Applied Mathematics, 2010, pp. 1373–1388.

[13] J. Correa and M. Wagner, LP-based online scheduling: From single to parallel machines, Math Program 119 (2007), 109–136.

[14] E. Feuerstein and L. Stougie, On-line single-server dial-a-ride problems, Theoret Comput Sci 268 (2001), 91–105.

[15] G. Goel and A. Mehta, Online budgeted matching in random input models with applications to adwords, Proc 19th Ann ACM-SIAM Symp Discr Algorithms, Society for Industrial and Applied Mathematics, 2008, pp. 982–991.

[16] M.X. Goemans, Improved approximation algorithms for scheduling with release dates, Proc 8th Ann ACM-SIAM Symp Discr Algorithms, Society for Industrial and Applied Mathematics, 1997, pp. 591–598.

[17] M.X. Goemans and J. Kleinberg, An improved approximation ratio for the minimum latency problem, Math Program 82 (1998), 111–124.

[18] M.X. Goemans, M. Queyranne, A.S. Schulz, M. Skutella, and Y. Wang, Single machine scheduling with release dates, SIAM J Discr Math 15 (2002), 165–192.

[19] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.R. Kan, Optimization and approximation in deterministic sequencing and scheduling: A survey, Ann Discr Math 5 (1979), 287–326.

[20] E. Günther, O. Maurer, N. Megow, and A. Wiese, A new approach to online scheduling: Approximating the optimal competitive ratio, Proc 24th Ann ACM-SIAM Symp Discr Algorithms, Society for Industrial and Applied Mathematics, 2013, pp. 118–128.

[21] L.A. Hall, A.S. Schulz, D.B. Shmoys, and J. Wein, Scheduling to minimize average completion time: Off-line and on-line approximation algorithms, Math Oper Res 22 (1997), 513–544.

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

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

[24] P. Jaillet and M.R. Wagner, "Online vehicle routing problems: A survey," The vehicle routing problem: Latest advances and new challenges, B. Golden, S. Raghavan, and E. Wasil (Editors), Springer US, 2008, pp. 221–237.

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

[26] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V.V. Vazirani, Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP, J ACM 50 (2003), 795–824.

[27] K. Jain, M. Mahdian, and A. Saberi, A new greedy approach for facility location problems, Proc 34th Ann ACM Symp Theory Comput, Association for Computing Machinery, 2002, pp. 731–740.

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

[29] S. Kim and P. Bobrowski, Impact of sequence-dependent setup time on job shop scheduling performance, Int J Production Res 32 (1994), 1503–1520.

[30] E. Koutsoupias and C.H. Papadimitriou, On the k-server conjecture, J ACM 42 (1995), 971–983.

[31] S.O. Krumke, W.E. de Paepe, D. Poensgen, and L. Stougie, News from the online traveling repairman, Theoret Comput Sci 295 (2003), 279–294.

[32] M. Lipmann, On-line routing problems, Ph.D. Thesis, Technische Universiteit Eindhoven, 2003.

[33] P. Liu and X. Lu, On-line scheduling of parallel machines to minimize total completion times, Comput Oper Res 36 (2009), 2647–2652.

[34] M. Mahdian, H. Nazerzadeh, and A. Saberi, Allocating online advertisement space with unreliable estimates, Proc 8th ACM Conference Electronic Commerce, Association for Computing Machinery, 2007, pp. 288–294.

[35] M. Mahdian and Q. Yan, Online bipartite matching with random arrivals: An approach based on strongly factor-revealing LPs, Proc 43rd Ann ACM Symp Theory Comput, Association for Computing Machinery, 2011, pp. 597–606.

[36] M. Mahdian, Y. Ye, and J. Zhang, Improved approximation algorithms for metric facility location problems, 5th Int Workshop, Approx 2002 Proc, 2002, pp. 229–242.

[37] N. Megow and A.S. Schulz, On-line scheduling to minimize average completion time revisited, Oper Res Lett 32 (2004), 485–490.

[38] N. Megow, M. Uetz, and T. Vredeveld, Models and algorithms for stochastic online scheduling, Math Oper Res 31 (2006), 513–525.

[39] V. Mirrokni, S. Oveis Gharan, and M. Zadimoghaddam, Simultaneous approximations for adversarial and stochastic online budgeted allocation, Proc 23rd Ann ACM-SIAM Symp Discr Algorithms, Society for Industrial and Applied Mathematics, 2012, pp. 1690–1701.

[40] E. Nowicki and S. Zdrzałka, A survey of results for sequencing problems with controllable processing times, Discr Appl Math 26 (1990), 271–287.

[41] C. Phillips, C. Stein, and J. Wein, "Scheduling jobs that arrive over time," Algorithms and data structures, S.G. Akl, F. Dehne, J.R. Sack, and N. Santoro (Editors), Springer Berlin Heidelberg, 1995, pp. 86–97.

[42] M.L. Pinedo, Scheduling: Theory, algorithms, and systems, Springer International Publishing, 2016.

[43] D. Shabtay and G. Steiner, A survey of scheduling with controllable processing times, Discr Appl Math 155 (2007), 1643–1666.

[44] D.B. Shmoys, J. Wein, and D.P. Williamson, Scheduling parallel machines on-line, SIAM J Comput 24 (1995), 1313–1331.

[45] V. Vinod and R. Sridharan, Dynamic job-shop scheduling with sequence-dependent setup times: Simulation modeling and analysis, Int J Advanced Manufacturing Technology 36 (2008), 355–372.

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

A On the Existence of ALG_l

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

A.1 A Nonexistence Example

Assume $m = 1$, $\mathbb{M}_1 = \mathbb{N}$, $O_1 = 1$, 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, $i \in \mathbb{N} \setminus \{1\}$, 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. Before further discussion, we first note that it is necessary for ALG_l to be the exact optimal solution. Otherwise, our analysis will not work due to Lemma 3.9 (even if ALG_l achieves an approximation ratio of $1 + \epsilon$).

The first assumption is about the metric space:

Assumption A.1 (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 Appendix A.1.

The second assumption is regarding the processing time:

Assumption A.2 (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 section is the following lemma:

Lemma A.3. *With Assumptions A.1 and A.2, 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 $\{ALG^i\}_{i=1}^{\infty}$ be a sequence of algorithms such that

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

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

Now let us consider the following sequence of n-dimensional real vectors: $S \triangleq \{(\min(c_1^{ALG^i}, t_l), \min(c_2^{ALG^i}, t_l), \ldots, \min(c_n^{ALG^i}, t_l))\}_{i=1}^{\infty}$. 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 $v_j \triangleq \lim_{i \to \infty} \min(c_j^{ALG^i}, t_l)$. It is sufficient to prove that there is an algorithm ALG' such that for all jobs $j \in [n]$, $\min(c_j^{ALG'}, t_l) \leq v_j$ 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 $\{ALG^i\}_{i=1}^{\infty}$ 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 $\{ALG^i\}_{i=1}^{\infty}$ 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 by 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 by $s^{j,i}$ the state of Machine 1 under which job $\rho(j)$ is processed under algorithm ALG^i. Because of the compactness (A.1) 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 allow 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 $j \in [n]$, $c^{ALG'}_{\rho(j)} \le \lim_{i\to\infty} c^{ALG^i}_{\rho(j)}$. To prove this, first note that $c^{ALG'}_{\rho(j)} = \sum_{i=1}^{j} d(\bar{s}^{j-1}, \bar{s}^j) + p_{1\rho(j)}(\bar{s}^j)$ and $c^{ALG^i}_{\rho(j)} \ge \sum_{i=1}^{j} d(s^{i,j-1}, s^{i,j}) + p_{1\rho(j)}(s^{i,j})$, 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 that $\{s^{j,i}\}_{i=1}^{\infty}$ converges to $\bar{s}^j$ and the lower-semicontinuity (A.2) assumption. □

B On the Necessity of Assumption 2.1

In this section, we show that it is necessary to impose Assumption 2.1 for our results to hold. In particular, we show the following proposition:

Proposition B.1. *When removing Assumption 2.1, for any positive value N, there exists a cost function such that the competitive ratio of PAC is at least N.*

Proof. Consider a variant of the online WTRP where each request j is associated with $\rho_j = (w_j, v_j)$ and the cost function is $\sum_{i=1}^{n} w_j \mathbb{1}(c^{ALG}_j \ge v_j)$, where $\mathbb{1}(c^{ALG}_j \ge v_j)$ is 1 when $c^{ALG}_j \ge v_j$ and 0 otherwise. Further let the metric space be the real line ($\mathbb{M} = \mathbb{R}$) and the depot be at 0 ($D = 0$). For any number $M > 0$, consider a problem instance $I(M)$ with $n = 3$ and

$$(r_j, l_j, w_j, v_j) = \begin{cases} (0.5, 0.5, 1, 0.5) & \text{for } j = 1, \\ (1, -1, 1, 1.5) & \text{for } j = 2, \\ (1, 1, M, 2) & \text{for } j = 3. \end{cases}$$

We first calculate $PAC(I(M))$. When the problem instance is I(M), $t_1 = 0.5$ and PAC completes request 1 at the first iteration and $c^{PAC}_1 = 1$. The second iteration is calculated at time $t_2 = 1.5$, and ALG_2 can complete only one of requests 2 and 3 before time 1.5. Completing request 2 has cost $w_2\mathbb{1}(1 \ge v_2) + w_3\mathbb{1}(1.5 \ge v_3) = 0$, and completing request 3 has cost $w_2\mathbb{1}(1.5 \ge v_2) + w_3\mathbb{1}(1 \ge v_3) = 1$, and hence ALG_2 completes request 2 before time 1.5. Therefore, PAC completes request 2 at the second iteration and $c^{PAC}_1 = 2.5$. Finally, PAC completes request 3 at the third iteration and $c^{PAC}_3 = t_3 + 1 = 5.5$. Overall, $PAC(I(M)) = 1 + 1 + M = M + 2$.

On the other hand, an offline algorithm could complete request 1 at time 0.5, request 3 at time 1, and request 2 at time 3, resulting in a total cost of 2, and thus $OPT(I) \le 2$.

Therefore, $PAC(I(2N-2))/OPT(I(2N-2)) \ge 2N/2 = N$, which completes the proof. □

C Proof of Corollary 3.3

Here we prove Corollary 3.3, which we repeat here:

Corollary 3.3. The competitive ratio of $PAC_{1,1}$ is at most $39/7 \approx 5.57$.

Proof. Using Theorem 3.1, it is sufficient to prove that for some positive integer N, the optimal objective value of $\text{LP}^{det}_{\alpha,\beta}(N)$ at $\alpha = \beta = 1$ is at most $39/7 \approx 5.57$. To have a simpler expression of (2d), we choose $N = 5$ so that when $i = 1$, $i = \left\lceil \frac{N-2i}{1+2\alpha} \right\rceil = \left\lfloor \frac{N-2i}{1+2\alpha} \right\rfloor$ =1.

Comparing (2a) and (2b), for all $i = 0, 1, \ldots, 4$, $\frac{(i+1)\alpha}{N}(T_i - T_{i+1}) \geq C_{i+1} - C_i \geq \frac{i\alpha}{N}(T_i - T_{i+1})$, and thus $T_i - T_{i+1} \geq 0$. Therefore, (2a) gives, for all $i = 1, 2, \ldots, 4$, $C_{i+1} - C_i \geq \frac{i\alpha}{N}(T_i - T_{i+1}) \geq \frac{\alpha}{N}(T_i - T_{i+1}) = \frac{1}{5}(T_i - T_{i+1})$. Taking the summation over all $i = 1, \ldots, 4$, we obtain $C_5 - C_1 \geq \frac{1}{5}(T_1 - T_5) = \frac{1}{5}T_1$, or equivalently,

$$C_1 + \frac{1}{5}T_1 \leq C_5. \tag{37}$$

Therefore,

$$\begin{aligned} C_5 + \frac{1}{5}T_0 &\leq C_1 + T_1 + C_1 + \frac{1}{5}T_1 && \text{((2d) with } i = 1) \\ &= 2C_1 + \frac{6}{5}T_1 \\ &\leq 6C_1 + \frac{6}{5}T_1 && (C_1 \geq 0) \\ &\leq 6C_5, && ((37)) \end{aligned}$$

or equivalently,

$$-5C_5 + \frac{1}{5}T_0 \leq 0. \tag{38}$$

Constraint (2c) gives

$$\frac{1}{2}C_5 + \frac{1}{6}T_0 \leq 1. \tag{39}$$

Multiplying both sides of (38) by $\frac{5}{14}$ and both sides of (39) by $\frac{39}{7}$, and then taking the summation, we obtain $C_5 + T_0 \leq \frac{39}{7}$, which implies the optimal objective value of $\mathrm{LP}^{det}_{\alpha,\beta}(N)$ at $\alpha = \beta = 1$ and $N = 5$ is at most $39/7 \approx 5.57$ and thus completes the proof. □

D Proof of Corollary 4.3

Here we prove Corollary 4.3, which we repeat here:

Corollary 4.3. The competitive ratio of PAC is at most $39/7 \approx 5.57$.

Proof. Using Theorem 4.1, it is sufficient to prove that the optimal objective value of $\mathrm{LP}^{det}(N)$ for some N is at most 39/7. To have simpler expressions of (30d) and (30e), we choose $N = 15$ so that when $i = 3$, $\frac{N-2i}{3}$, $\frac{N}{3}$, and $\frac{2i}{3}$ are integers.

Constraint (30e) with $i = 3$ gives

$$X_{15} + T_{0,5} \leq X_3 + T_{3,15} + X_3 + T_{0,2} + T_{3,3}. \tag{40}$$

The above inequality involves X_3 and X_{15}. In what follows, we derive a lower bound for $X_{15} - X_3$. Summing over (30b) for $i = 3, 4, \ldots, 14$, we obtain

$$\begin{aligned} X_{15} - X_3 \geq & \sum_{i=3}^{14} T_{i,i} - T_{i+1,i} \\ = & T_{3,3} + \sum_{i=3}^{13} (T_{i+1,i+1} - T_{i+1,i}) - T_{15,14} \\ = & T_{3,3} + \sum_{i=3}^{13} (T_{i+1,i+1} - T_{i+1,i}) . \qquad (T_{15,14} = 0) \end{aligned} \tag{41}$$

In what follows, we show for all $i = 3, \ldots, 13$, $T_{i+1,i+1} - T_{i+1,i} \geq 0$. Comparing (30b) and (30c), for all $i = 0, 1, \ldots, 14$, $T_{i,i+1} - T_{i+1,i+1} \geq X_{i+1} - X_i \geq T_{i,i} - T_{i+1,i}$, or equivalently, $T_{i,i+1} - T_{i,i} \geq T_{i+1,i+1} - T_{i+1,i}$. Constraints (30a) with $i = 1, 2, \ldots, 14$ and $j = i$ give $T_{i,i+1} + T_{i,i-1} \leq 2T_{i,i}$, or equivalently, $T_{i,i} - T_{i,i-1} \geq T_{i,i+1} - T_{i,i}$. Combining the two inequalities above, we obtain, for all $i = 1, \ldots, 14$, $T_{i,i} - T_{i,i-1} \geq T_{i+1,i+1} - T_{i+1,i}$. Using this inequality repeatedly, for all $i = 1, \ldots, 14$, $T_{i,i} - T_{i,i-1} \geq T_{15,15} - T_{15,14} = 0$. Hence, for all $i = 3, \ldots, 13$, $T_{i+1,i+1} - T_{i+1,i} \geq 0$. As a result, (41) gives $X_{15} - X_3 \geq T_{3,3}$, or equivalently,

$$X_3 + T_{3,3} \leq X_{15}. \tag{42}$$

Inequality (40) motivates us to prove the following three inequalities based on Constraints (30a). Constraints (30a) with $i = 3$ gives, for all $j = 1, \ldots, 14$, $T_{3,j-1} - 2T_{3,j} + T_{3,j+1} \leq 0$. As a result,

$$\begin{aligned} & -5T_{3,3} + T_{3,15} \\ = & 4T_{3,0} - 5T_{3,3} + T_{3,15} \qquad (T_{3,0} = 0) \\ = & \sum_{j=1}^{3} 4j(T_{3,j-1} - 2T_{3,j} + T_{3,j+1}) + \sum_{j=4}^{14} (15 - j)(T_{3,j-1} - 2T_{3,j} + T_{3,j+1}) \leq 0, \end{aligned}$$

or equivalently,

$$T_{3,15} \leq 5T_{3,3}. \tag{43}$$

Similarly, when $i = 0$, for all $j = 1, \ldots, 14$, $T_{0,j-1} - 2T_{0,j} + T_{0,j+1} \leq 0$. As a result,

$$\begin{aligned} & 3T_{0,15} + 10T_{0,2} - 13T_{0,5} \\ = & \sum_{j=3}^{5} 10(j - 2)(T_{0,j-1} - 2T_{0,j} + T_{0,j+1}) + \sum_{j=6}^{14} 3(15 - j)(T_{0,j-1} - 2T_{0,j} + T_{0,j+1}) \leq 0. \end{aligned} \tag{44}$$

In addition,

$$\begin{aligned} & T_{0,15} - 3T_{0,5} \\ = & T_{0,15} - 3T_{0,5} + 2T_{0,0} \qquad (T_{0,0} = 0) \\ = & \sum_{j=1}^{5} 2j(T_{3,j-1} - 2T_{3,j} + T_{3,j+1}) + \sum_{j=6}^{14} (15 - j)(T_{3,j-1} - 2T_{3,j} + T_{3,j+1}) \leq 0. \end{aligned} \tag{45}$$

Going back to (40), we have

$$\begin{aligned} X_{15} + T_{0,5} \leq & X_3 + T_{3,15} + X_3 + T_{0,2} + T_{3,3} && \text{((40))} \\ \leq & 2X_3 + 6T_{3,3} + T_{0,2} && \text{((43))} \\ \leq & 6(X_3 + T_{3,3}) + T_{0,2} && (X_3 \geq 0) \\ \leq & 6X_{15} + T_{0,2}, && \text{((42))} \end{aligned}$$

or equivalently,

$$-5X_{15} + T_{0,5} - T_{0,2} \leq 0. \tag{46}$$

From (30d), we have

$$T_{0,5} + X_{15} \leq 2. \tag{47}$$

Multiplying both sides of (46) by 10, (47) by 78, (44) by 1, and (45) by 25, and taking the summation, we have $28X_{15} + 28T_{0,15} \leq 156$, which means $X_{15} + T_{0,15} \leq 156/28 = 39/7$. Therefore, the optimal objective value of $\text{LP}^{det}(N)$ with $N = 15$ is at most 39/7, which concludes the proof. □