Online weighted throughput maximization scheduling with a busy-time budget

Jiefei Xue

Abstract

In this paper, we consider an online scheduling problem, *weighted throughput maximization scheduling with a busy-time budget*. In this problem, jobs are released with released times r_j, processing times p_j, and deadlines $d_j = r_j + p_j$. Moreover, jobs have weights w_j depending on the setting. In the *proportional setting*, the weight of a job equals to the length of this job. While in the *categorized setting*, a job has weight 1 if the length of this job is less than the given threshold ω, otherwise it has weight 2. We are also given machines, and each machine can run g jobs simultaneously. A busy-time budget T is also given. The objective is to gain as much weight as possible by scheduling jobs on machines using busy-time at most T. For infeasible instances of proportional setting, we consider general, clique, and one-sided clique instances, and prove that there is no deterministic online algorithm with a constant competitive ratio, and the greedy algorithm is an optimal online algorithm. For infeasible instances of categorized setting, we consider general, clique, and one-sided clique instances, and prove that no deterministic online algorithm has a constant competitive ratio unless it is a one-sided clique instance, and T is 2. For feasible one-sided clique instances of categorized setting, we design an adversary and prove that no deterministic online algorithm has a competitive ratio lower than $\frac{9}{7}$.

1 Introduction

With the rapid development of optical networks and data centers, the consumption of energy is growing significantly[1]. Thus, achieving higher energy efficiency has become one of the most concerned topics[2], which is not only environmentally friendly but also beneficial to lower the energy cost or increase profit. From the perspective of a data center, when a certain amount of resources or budget is given, better energy efficiency can be achieved by scheduling jobs properly to obtain a higher profit[3, 4]. The profit of jobs can be weighted in different ways, such as the amount of time consumed or priority. Furthermore, in reality, a data center usually has no information about upcoming jobs. It knows the information about the job it needs to process only when it is released. Therefore, the scheduling problem can be modeled as an online problem. There are many online algorithm research on energy field[5, 6, 7, 8, 9, 10]. In this paper, we propose and discuss the *online weighted throughput maximization scheduling with a busy-time budget*, which is one of the most related and representative machine scheduling problems that can help achieve higher energy efficiency when processing jobs[1, 11].

Machine scheduling problem has been studied for many years[12]. The basic concept of machine scheduling problems is scheduling jobs on machines to achieve some objectives such as minimizing the cost or maximizing the gain. We need to process jobs by scheduling them on machines. The jobs are often characterized by $\boldsymbol{r_j}$(**release time**): the time when the job is known, $\boldsymbol{p_j}$(**processing time**): the amount of time needed to finish the job, and $\boldsymbol{d_j}$(**deadline**): the time before which the job must be finished. Many different variants of machine scheduling problems have been proposed according to different objectives and settings(see survey[13]). The closest related work to our problem was studied by Shalom et al.[11], where they proposed and discussed the online unweighted throughput maximization problem. In this problem, every job brings the same amount of profit, no matter how long the job is. But in reality, jobs could have different weights depending on, for instance, the length or the priority of the job. Thus, based on their work, we further propose and analyze the proportional weighted throughput maximization problem and the categorized weighted throughput maximization problem.

Instances	Lower bound	Upper bound	Optimal algorithm
General	gT	gT	Greedy
Clique	gT/2	gT/2	Greedy
One-sided clique	T	T	Greedy

Table 1: Results for proportional throughput maximization problem

Instances	Lower bound
Infeasible General	$2g$, when T≤ 4 $gT/2$, when T>4
Infeasible Clique	$2g$, when $T > \omega \geq 2$ and $8 \geq T \geq 3$ $gT/4$, when $T > \omega \geq 2$, and $T > 8$, $2g$, when $T = \omega = 3$ $g\lfloor \frac{T}{3} \rfloor$, when $T = \omega > 3$
Infeasible One-sided clique	$\frac{gT}{g+1}$, when $T \geq \frac{2}{g} + 2$ 2, when $2 \leq T < \frac{2}{g} + 2$
Feasible One-sided clique	$\frac{4}{3}$, when $T < 2\omega - 1$ $\frac{9}{7}$, when $T \geq 2\omega - 1$

Table 2: Results for categorized throughput maximization problem

Instances	Extra g processors	Extra T busy-time budget
Infeasible general	g, when $2 \leq T \leq 4$ $gT/4$, when $T > 4$	$gT/2$, when $T \geq 4$ $2g$, when $2 \leq T < 4$
Infeasible one-sided clique	$\frac{gT}{2g-1}$, when $T \geq 3$	does not help

Table 3: Competitive ratio lower bounds for resource augmentation

The structure of this paper is as follows. In Chapter 2, we introduced domain knowledge of machine scheduling problems and summarized the results of literature on related machine scheduling problems. In Chapter 3, we described some frequently used concepts and propositions in the analysis. In Chapter 4, we introduced the **Proportional Weighted Throughput Maximization Problem** and gave both negative results and positive results on infeasible instances by analysis. Chapter 5 is the most significant part of this paper, which introduced the **Categorized Weighted Throughput Maximization Problem** and gave both negative and positive results on not only infeasible instances but also feasible instances. In Chapter 6, we extended the **Categorized Weighted Throughput Maximization Problem** to the case where resource augmentation is allowed, and both negative results and positive results on infeasible instances are given. In Chapter 7, we summarized the contributions of this paper and proposed some possible future research problems based on this paper.

2 Literature review

2.1 Machine scheduling and terminologies

Before we start reviewing the literature, there are some concepts that need to be introduced for better expression.

Job flexibility: Depending on the length of the feasible interval $[r_j, d_j)$ and processing time p_j, jobs can have different flexibility, which is *unit*, *rigid* or *flexible*. A rigid job occupies the machine during the whole time interval. But a flexible job only occupies the machine for p_j time from its start time s_j and $[s_j, s_j + p_j) \in [r_j, d_j)$.

Special instances: In this paper, we consider general instances and some special instances, *clique instances*, *one-sided clique instances*, and *feasible instances*. *One-sided clique instances*, where all jobs

have the same release time or deadline. *Clique instances*, where all jobs share the same common time t within the available interval between their release times and deadlines. *Feasible instances*, where there exists an offline algorithm that can schedule all released jobs.

Machine types: machine can be identical, uniform, or non-related. On *identical machines*, any job can be scheduled on any machine, and the amount of time needed to process this job is the same for any machine. On *uniform machines*, jobs can have j types and machines can have m types, and any job can be scheduled on any machine. However, the processing time of job j on machine m depends on the speed s_{mj}. On *related machines*, the amount of time needed to process any job on any machine does not depend on the job type or the machine type.

Machine parallelism: In cases without a parallelism setting, there is only one job that can be processed during a time interval. But in cases with parallelism settings, any machine has g processors, and there are at most g jobs that can be processed simultaneously.

Preemption and non-preemption: in non-preemptive cases, a job can not be interrupted after it is started. But in preemptive cases, a job can be interrupted and can be continued on the same or another machine.

2.2 Related results

2.2.1 Makespan minimization

Makespan minimization problem is a very classic, important, and extensively studied topic in machine scheduling field[14, 15]. In this problem, we are given jobs with released times r_j and processing times p_j. We are given m machines to schedule jobs. At any time, any machine can only process at most one job, and no job can be processed on multiple machines at the same time. We need to schedule all released jobs on machines so that all jobs can be finished before their deadlines and no job can be dropped.

The objective is to schedule all jobs on machines and minimize the latest completion time(the time when a job is finished) of all jobs.

Albers et al.[16] studied the online makespan problem where parallelism is allowed on machines and they developed an algorithm with a competitive ratio of $4/3 + \epsilon$, for any $0 < \epsilon \leq 1$, which uses a constant number of schedules, and the constant number equals to $1/\epsilon^{O(log(1/\epsilon))}$.

2.2.2 Machine minimization

In this problem, we are given jobs with released times r_j, deadlines d_j and processing times p_j. We are given machines to schedule jobs. At any time, any machine can only process at most one job, and no job can be processed on multiple machines at the same time. We need to schedule all released jobs on machines so that all jobs can be finished before their deadlines and no job can be dropped. For each job, we can either schedule it on an existing machine or schedule it on a new machine.

The objective is to schedule all jobs on machines and minimize the number of machines used.

In[17], chen et al. considered preemptive jobs and proposed a $O(log\ m)$-competitive online algorithm, where m is the minimum number of machines used in an offline optimal solution. And their work is the first improvement on the previously best-known result proposed by Phillips et al. in 1997[18], which is a $O(log(p_{max}/p_{min}))$-competitive algorithm, where p_{max} and p_{min} are the maximum and minimum job lengths, respectively.

2.2.3 Busy time minimization

In the online parallel machine scheduling field, much attention is paid to the busy time minimization problem[13].

In this problem, we are given jobs with released times r_j, deadlines d_j and processing times p_j. We are given machines and each machine can run g jobs simultaneously. We need to schedule all released jobs on machines so that all jobs can be finished before their deadlines and no job can be dropped. Let $busy = [s, t)$ be the time interval during which there is at least one job being processed on a machine. And the length of a busy time interval $len(busy) = t - s$. Let $BUSY_m$ be the total busy time interval on machine m, which is the set of all $busy$ on machine m. Furthermore, let $len(BUSY_m)$ be the total length of all busy time intervals on machine m, which is the sum of all $len(busy)$ on machine m.

The objective is to schedule all jobs on machines and minimize the total length of all busy time intervals over all machines $\sum_{m=0}^{\infty} len(BUSY_m)$.

Different variants are proposed according to different job flexibility, job assumptions, preemption and clairvoyant.

For the **rigid** job setting, Shalom et al.[11] showed that no online deterministic algorithm can have a competitive ratio better than g-competitive(parallelism parameter). And they proved this lower bound by using a delicate adversary. It first releases a very long job to force a machine to run for a long period of time and then releases jobs one by one, and each job is longer than the previous one so that the machine has to schedule the shorter jobs before scheduling longer jobs, which can not fully make use of the parallelism feature. When all available processors on a machine are occupied, a new machine is used to schedule the longest job, which again forces a machine to run for a long time. And when a new machine is used, the adversary releases jobs with a later release time such that they can not be run simultaneously with the scheduled jobs.

They also proposed a $5logp_{max}$- competitive online algorithm, where p_{max} is the longest processing time. The intuitive idea of this algorithm is dividing jobs into different buckets according to their lengths and only jobs of the same bucket can be scheduled on the same machine. Therefore, a scheduled job will only prevent jobs of similar length from being scheduled on the same machine rather than much longer jobs. Furthermore, they studied one-sided clique instances and clique instances, and respectively proposed algorithm has a competitive ratios of $1+\varphi$ and $2(1+\varphi)$ where $\varphi = \frac{1+\sqrt{5}}{2}$ is the **Golden Ratio**. The intuitive idea of the algorithm for one-sided clique instances is dividing jobs into buckets according to their lengths and scheduling g jobs from the same bucket on the same machine to make user jobs would be overlapping with jobs of similar length rather much shorter or longer jobs.

As for the **flexible** job setting, Koehler ti al.[19] proposed a 5-competitive online deterministic algorithm when g is not bounded and a $log\frac{p_{max}}{p_{min}}$-competitive algorithm when g is bounded. Furthermore, they studied a semi-online setting where the algorithm knows the future by $2p_{max}$ units of time and got a constant competitive ratio of 12.

2.2.4 Throughput maximization

In this problem, we are given jobs with released times r_j, deadlines d_j, processing times p_j, and weights w_j. We are given machines and each machine can run g jobs simultaneously. Scheduling a job with weight w_j on a machine brings profit w_j. We are also given a busy-time budget T. We need to schedule jobs on machines to gain as much profit as possible. Let $busy = [s, t)$ be the time interval during which there is at least one job being processed on a machine. The length of a busy time interval $len(busy) = t - s$. Let $BUSY_m$ be the total busy time interval on machine m, which is the set of all $busy$ on machine m. Furthermore, let $len(BUSY_m)$ be the total length of all busy time intervals on machine m, which is the sum of all $len(busy)$ on machine m.

The objective is to maximize the obtained profit by scheduling jobs on machines using busy-time at most T, which means the total length of all busy time interval over all machines $\sum_{m=0}^{\infty} len(BUSY_m) \leq T$.

In this problem, we do not need to schedule all released jobs on machines, which means that we can drop some jobs, except for the feasible instances. In this case, the online algorithm might drop a job because there is no enough busy-time budget left for this job, or dropping this job might allow the online algorithm to schedule the upcoming jobs with greater profit.

One of the closest relevant studies is studied by Shalom et al. In [11], they proved that no online algorithm has a competitive ratio better than gT and there exists an algorithm that schedules every available job on any machine has a competitive ratio of exactly gT. However, gT-competitive indicates a really bad performance. To obtain useful results, they focused on feasible one-sided clique instances and they not only showed that any online algorithm has a competitive ratio of at least $2 - \frac{2}{g+1}$ but also proposed an online algorithm, which accepts sufficient numbers of short jobs and then accept longer jobs if they do not occupy too much remaining time, with a competitive ratio depending on g but at most $9/2$.

2.2.5 Resource augmentation

Resource augmentation is a widely used and accepted methodology to compare the performance of objective algorithm with the performance candidate algorithm, which can be seen from a survey of

scheduling problems with many resource augmentation results[20]. It was introduced by Kalyanasundaram and Pruhs in [21] where they compared the optimal algorithm with a unit speed machine to their candidate algorithm equipped with a faster machine. And in [18], Phillips et al. defined a approximation algorithm which can achieve an objective function value at most $\rho \cdot OPT$ using a machine of speed s, where OPT is the optimal objective function value using one machine of unit speed. And they named this methodology resource augmentation analysis. Up till now, resource augmentation is still a very popular and powerful methodology. One of the latest research using resource augmentation is [22], where bansal et al. discussed serval non-preemptive min-sum scheduling problems and proposed the first $O(1)$-speed $O(1)$-approximation polynomial-time algorithms.

3 Preliminaries

Some concepts are mentioned or used frequently in this paper.

The maximum length of a job is T because a job of length greater than T cannot be scheduled by any solution when the given time budget is T. The ratio between the total weight and total length of scheduled jobs in a busy time-interval of one processor on one machine is called **weight density**. We consider **feasible** and **infeasible** instances in this paper. An input instance is **feasible** if there exists a schedule that can assign all the jobs in this instance with total busy time at most T. An input instance is **infeasible** if there is no schedule that can assign all the jobs in this instance with total busy time at most T.

A machine is **busy** at time t if there is at least one job being processed. A **time-slot** t on machine m is of length 1 and denoted by a tuple (m, t). A **time-interval** on machine m is a set of time-slot on machine m.

In this paper, we often use the greedy algorithm to analyze the competitive ratio upper bound of instances. All the `greedy algorithm` mentioned in the following section always schedules a released job without exceeding the busy-time budget by scheduling it on a currently available machine, in machine opening order, or open a new machine when there is no available machine.

All problems mentioned in the paper are online-list problems. The time when a job is released does not equal to its release time. Jobs with later release time might be released earlier.

The following propositions are used in this paper.

Proposition 1 *For any instances, any deterministic online algorithm has to schedule the first released job.*

Proof. For any instance, if a deterministic online algorithm ALG does not schedule the first released job. Then, the adversary can stop releasing job and the first released job becomes the only released job. Moreover, an optimal solution will schedule this job and gains the weight of this job while ALG gains 0 profit. Hence, the competitive ratio is unbounded. Therefore, **Proposition 1** is proven.

Proposition 2 *Assume the minimum length of a job is 1, the maximum length of a job is T. Release times and deadlines are integral.*

Proposition 3 *Greedy algorithm has throughput 0 only when there is no job released, and in this case, the optimal solution also has throughput 0.*

4 Proportional Weighted Throughput Maximization Problem

In this section, we consider the **Busy Time Weighted Throughput Maximization Problem** where the weight of a job equals to its length. W.l.o.g, we assume the minimum length of a job is 1. Release times and deadlines are integral.

Lemma 1 *For any instance of **Proportional Weighted Throughput Maximization Problem**, any algorithm can gain Throughput at most gT .*

Proof. For any instance, the maximum WEIGHT DENSITY of a busy time-slot on a single processor is 1 and there are g processors on one machine which means at most g jobs can be scheduled in a busy time-slot on one machine. Thus, gaining $g+1$ throughput in a single time-slot requires at least two machines, which takes two busy-time slot. Therefore, for any instance, one busy time-slot can gain at most g throughput. Moreover, the maximum number of unit busy-time slots is T because the time budget is T. Hence, for any instance, an algorithm can gain at most gT Throughput.

4.1 Infeasible instances

In this section, we consider infeasible instances. We show that for the **Proportional Weighted Throughput Maximization Problem**, there is no deterministic constant competitive algorithm against infeasible instances.

4.1.1 Infeasible general instances

Theorem 1 *For infeasible general instances of **Proportional Weighted Throughput Maximization Problem**, no deterministic online algorithm has a competitive ratio lower than gT.*

Proof. Consider an adversary that first releases a job of length 1 at $[0,1)$. Any online algorithm ALG has to schedule this job according to **Proposition 1** and consume 1 unit of busy-time budget. Then, the adversary releases g jobs of length T at $[1,T+1)$. Since ALG already consumed 1 unit of busy-time budget when scheduling a job of length 1 at $[0,1)$, the remaining $T-1$ busy-time budget is not enough to schedule a job of length T at $[1,T+1)$. Hence, ALG cannot schedule any of these jobs, and thus $tput^A \leq 1$. However, an optimal solution will schedule only these jobs and drop the first job of length 1 and $tput^* \geq gT$. As a result, the competitive ratio is gT.

■

Theorem 2 *For infeasible general instances of **Proportional Weighted Throughput Maximization Problem**, the `greedy algorithm` is an optimal deterministic online algorithm.*

Proof. The `greedy algorithm` can gain throughput at least 1 because the minimum job length is 1 and it can schedule at least the first released job. While the optimal solution can gain throughput at most gT according to **Lemma 1**. As a result, the competitive ratio of the `greedy algorithm` is at most gT which matches with the lower bound. Hence, **Theorem 2** is proven.

■

4.1.2 Infeasible clique instances

Theorem 3 *For infeasible clique instances of **Proportional Weighted Throughput Maximization Problem**, no online algorithm has a competitive ratio lower than $gT/2$*

Proof. Consider an adversary that first releases a job of length 2 at $[0,2)$. Any online algorithm ALG has to schedule this job according to **Proposition 1** and consume 2 units of busy-time budget. Then, the adversary releases g jobs of length T at $[1,T+1)$. Since ALG already consumed 2 units of busy-time budget when scheduling a job of length 2 at $[0,2)$ therefore the remaining $T-2$ busy-time budget is not enough to schedule a job of length T at $[1,T+1)$. Hence, ALG cannot schedule any of these jobs, and thus $tput^A \leq 2$. However, an optimal solution will schedule only these jobs and drop the first job of length 2 and $tput^* \geq gT$. As a result, the competitive ratio is at least $gT/2$.

■

Theorem 4 *For infeasible clique instances of **Proportional Weighted Throughput Maximization Problem**, the `greedy algorithm` is an optimal deterministic online algorithm.*

Proof. For clique instances, the maximum throughput an algorithm could obtain is gT according to **Lemma 1**. When the greedy algorithm obtain throughput $tput^A \geq 2$, the competitive ratio is less than $gT/2$ which is bounded by $gT/2$. Thus, the remaining case is when $tput^A < 2$. Furthermore, since all jobs are integral, we only need to consider the case when the `greedy algorithm` gains throughput 0 or 1.

When the `greedy algorithm` gains throughput 0, there is no available job and any algorithm has throughput 0. Otherwise, the `greedy algorithm` will schedule at least one job.

When the `greedy algorithm` gains throughput 1, there must be only one job of length 1 released and both the optimal solution and the greedy solution will schedule this job, and thus the competitive ratio is 1. Otherwise, if there is more than one job released, assume the `greedy algorithm` gains throughput 1, the first released job must be a job of length 1, and the `greedy algorithm` scheduled this job. However, in any clique instance, all jobs share at least one time-slot, therefore the first job of length 1 will be contained by the second released job, and since its length is at most T, it means that the second job is available to the `greedy algorithm`. As a result, the `greedy algorithm` will schedule this job and have integral throughput greater than 1. Combining with **Lemma 1**, the upper bound of the competitive ratio for the `greedy algorithm` is $gT/2$ which matches with the lower bound. Hence, **Theorem 4** is proven.

■

4.1.3 Infeasible one-sided clique instances

Theorem 5 *For infeasible one-sided clique instances of* ***Proportional Weighted Throughput Maximization Problem****, no deterministic online algorithm has a competitive ratio lower than T.*

Proof. Consider an adversary that first releases a job of length 1 at $[0, 1)$. Any online algorithmALG has to schedule this job according to **Proposition 1** and consume 1 unit of busy-time. Then, the adversary keeps releasing jobs of length 1 at $[0, 1)$ until gT jobs are released or a machine is filled with g jobs, or a new machine is used. We consider 2 cases.

Case 1. There is one machine used by ALG and it is not filled with g jobs at $[0, 1)$. Then, there are at most $g - 1$ jobs of 1 scheduled on this machine, thus $tput^A \leq g - 1$. But an optimal solution will schedule all gT jobs of 1 on T machines and $tput^* \geq gT$. So the competitive ratio is at least $\frac{gT}{g-1}$.

Case 2. There is one machine used by ALG and it is filled with g jobs at $[0, 1)$ or two machines are used. Then, the adversary releases g jobs of length T at $[0, T)$. In this case, ALG cannot schedule any of these jobs on any machine otherwise, the total busy-time on all machines will exceed the given busy-time budget. Therefore $tput^A \leq g$. However, an optimal solution will only schedule these jobs and $tput^* \geq gT$. As a result, the competitive ratio is at least T.

■

Theorem 6 *For infeasible one-sided clique instances of* ***Proportional Weighted Throughput Maximization Problem****, the* `greedy algorithm` *is an optimal online deterministic algorithm.*

Proof. Since the minimum job length is 1, the minimum throughput one processor can gain is 1. Moreover, there are g processors on a machine, and jobs in a one-sided clique instance share the same release time (deadline) so the `greedy algorithm` could at least gain throughput g, which is $tput^A \geq 1$. Combining with **Lemma 1**, we know the competitive ratio of the `greedy algorithm` is at most T which matches with the lower bound. As a result, **Theorem 6** is proven.

■

5 Categorized Weighted Throughput Maximization Problem

In this section, we consider the **Categorized Weighted Throughput Maximization Problem** where the weight of a job depends on whether its length is greater than or equal to a given threshold $\omega \geq 2$. If a job has length $\geq \omega$ then, its weight is 2, otherwise, its weight is 1. The threshold is less than or equal to the given time budget, $\omega \leq T$. otherwise, all jobs will be shorter than ω and have weights 1.

The objective is to maximize the obtained throughput by scheduling jobs, consuming busy time less than or equal to the given time budget.

Lemma 2 *For any instance of the* ***Categorized Weighted Throughput Maximization Problem****, any algorithm can gain throughput at most gT.*

Proof. For any instance in **Categorized Weighted Throughput Maximization Problem**, since the maximum weight of a job is 2 and the threshold $\omega \geq 2$, WEIGHT DENSITY is at most 1. And the maximum length of busy time-intervals of one processor is T because the time budget is T. Thus, for a single processor, the maximum throughput is T. Furthermore, since the maximum number of processors on one machine is g, any algorithm can gain throughput at most gT.

■

5.1 Infeasible instances

In this section, we consider infeasible instances. We show that for the **Categorized Weighted Throughput Maximization Problem**, there is no deterministic competitive algorithm against infeasible instances.

5.1.1 Infeasible general instances

Theorem 7 *For infeasible general instances of* ***Categorized Weighted Throughput Maximization Problem****, no deterministic online algorithm has a competitive ratio lower than* $2g$*, when* $T \leq 4$*, or* $\frac{gT}{2}$*, when* $T > 4$*.*

Proof. Consider the following adversary:

when $T \leq 4$, it first releases a job of length 1 at $[0, 1)$. Any online algorithm ALG has to schedule this job according to **Proposition 1** and consume 1 unit of busy-time budget. Then, the adversary releases g jobs of length T at $[1, T + 1)$. Since ALG already consumed 1 unit of busy-time budget when scheduling a job of length 1 at $[0, 1)$, therefore the remaining $T - 1$ units of busy-time budget is not enough for scheduling a job of length T at $[1, T + 1)$ on any machine. Hence, ALG cannot schedule any of these jobs, and thus online algorithm can gain throughput $tput^A \leq 1$. However, an optimal solution will schedule only these jobs and drop the first job of length 1 and gain throughput $tput^* \geq 2g$. Thus, the competitive ratio is at least $2g$.

When $T > 4$, it first releases a job of length T at $[0, T)$. Any online algorithm ALG has to schedule this job and consume T units of busy-time budget. Then, the adversary releases gT jobs of length 1 at $[T, T + 1)$. Since ALG already consumed T units of busy-time budget when scheduling a job of length T at $[0, T)$, therefore there is no more busy-time budget for scheduling a job of length 1 at $[T, T + 1)$. Hence, ALG cannot schedule any of these jobs, and thus it gains throughput $tput^A \leq 2$. However, an optimal solution will schedule only these jobs and drop the first job of length T and gain throughput $tput^* \geq gT$. Thus, the competitive ratio is at least $\frac{gT}{2}$.

As a result, **Theorem 7** is proven.

■

Theorem 8 *For infeasible general instances of* ***Categorized Weighted Throughput Maximization Problem****, when the* `greedy algorithm` *gains throughput at least 2, it is an optimal online deterministic algorithm.*

Proof. According to **Lemma 2**, we know an optimal solution can gain throughput at most gT. Then, we consider 4 cases, the `greedy algorithm` gains throughput greater than or equal to 2, equals to 1.

Case 1: When the `greedy algorithm` gains throughput greater than or equal to 2, the competitive ratio is lower than or equal to $gT/2$, which matches with the lower bound of the competitive ratio when $T > 4$. Then we still need to prove that when $2 \leq T \leq 4$, the upper bound of the competitive ratio matches with the lower bound, which is $2g$. Since $tput^* \leq gT$, if the upper bound of the competitive ratio matches with the lower bound $2g$, then, the upper bound of the competitive ratio is at most $\frac{gT}{\frac{T}{2}} \leq 2g$. Hence, the `greedy algorithm` must gain throughput $tput^A \geq \frac{T}{2}$. Since $2 \leq T \leq 4$, $\frac{T}{2}$ is at most 2. And **Case 1** is the case when the `greedy algorithm` gains throughput greater than or equal to 2. Hence, the `greedy algorithm` already satisfied the condition that $tput^A \geq \frac{T}{2}$ when $2 \leq T \leq 4$. So when $2 \leq T \leq 4$, the upper bound of the competitive ratio of the `greedy algorithm` is at most $2g$, which matches with the lower bound when $2 \leq T \leq 4$.

As a result, **Theorem 8** is proven.

■

5.1.2 Infeasible clique instances

Theorem 9 *For infeasible clique instances of* ***Categorized Weighted Throughput Maximization Problem****, no deterministic online algorithm has a competitive ratio lower than the following lower bounds:*
(1) when $T > \omega \geq 2$ *and* $8 \geq T \geq 3$*, the competitive ratio is at least* $2g$
(2) when $T > \omega \geq 2$*, and* $T > 8$*, the competitive ratio is at least* $\frac{gT}{4}$
(3) when $T = \omega = 3$*, the competitive ratio is at least* $2g$
(4) when $T = \omega > 3$*, the competitive ratio is at least* $g\lfloor \frac{T}{3} \rfloor$*.*

Proof. Consider the following adversary:

Case 1: The adversary first releases a job of length 2 at $[0, 2)$. Any online algorithm ALG has to schedule this job according to **Proposition 1** and consume 2 units of busy-time budget. Then, the adversary releases g jobs of length T at $[1, T+1)$. Since ALG already consumed 2 units of busy-time budget when scheduling a job of length 2 at $[0, 2)$, there is no busy-time budget left for opening a new machine to schedule a job of length T at $[1, T+1)$. Furthermore, ALG cannot schedule any of these jobs on the machine with a job of length 2. Otherwise, the total busy-time will be $T+1$, which is greater than the given time budget T. However, an optimal solution will schedule only jobs of length T and drop the first job of length 2. In this case, $tput^* \geq 2g$. While ALG gains throughput $tput^A \leq 2$ when $\omega = 2$, or $tput^A \leq 1$ when $\omega \geq 3$.

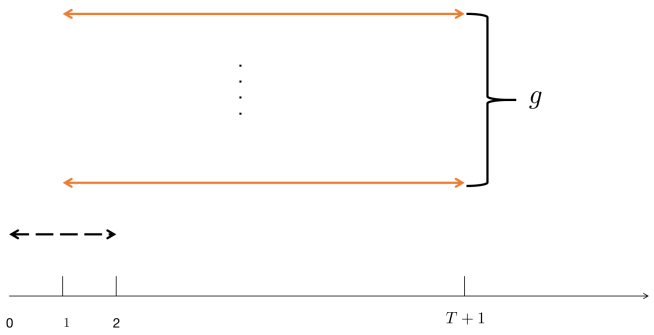

Figure 1: Theorem 9, case 1 (dotted lines are online schedule, solid lines are optimal schedule)

As a result, the competitive ratio is at least g when $\omega = 2$, or $2g$ when $\omega \geq 3$, corresponding to lower bound (1).

Case 2: The adversary first releases a job of length T at $[0, T)$. Any online algorithm ALG has to schedule this job according to **Proposition 1** and consume T units of busy-time budget. Then, the adversary releases $\frac{gT}{2}$ jobs of length 2 at $[T-1, T+1)$. Since ALG already consumed T units of busy-time budget when scheduling a job of length T at $[0, T)$, there is no more busy-time budget for opening a new machine to schedule a job of length 2 at $[T-1, T+1)$. Furthermore, ALG cannot schedule any of these jobs on the machine with a job of length T. Otherwise, the total busy-time will be $T+1$, which is greater than the given time budget T. However, an optimal solution will schedule only jobs of length 2 and drop the first job of length T. In this case, $tput^* \geq gT$ when $\omega = 2$, or $tput^* \geq \frac{gT}{2}$ when $\omega \geq 3$. While ALG gains throughput $tput^A \leq 2$.

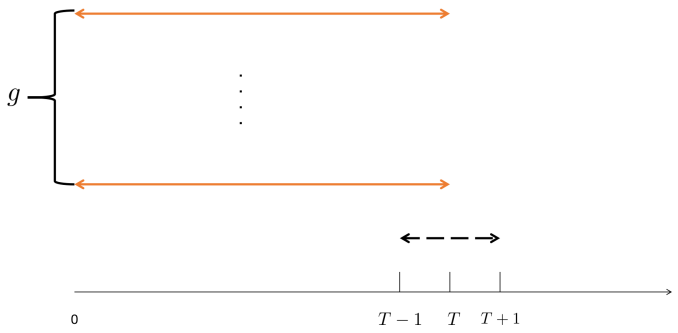

Figure 2: Theorem 9, case 2 (dotted lines are online schedule, solid lines are optimal schedule)

As a result, the competitive ratio is at least $\frac{gT}{2}$ when $\omega = 2$, or $\frac{gT}{4}$ when $\omega \geq 3$, corresponding to lower bound (2).

Case 3: **When $T = \omega$**, the adversary first releases a job of length $\omega - 1$ at $[0, \omega - 1)$. Any online algorithm ALG has to schedule this job according to **Proposition 1** and consume $\omega - 1$ units of busy-time budget. Then, the adversary releases $g\lfloor \frac{T}{T-\omega+3} \rfloor$ jobs of length $T - \omega + 3$ at $[\omega - 2, T + 1)$. Since ALG already consumed $\omega - 1$ units of busy-time budget when scheduling a job of length $\omega - 1$ at $[0, \omega - 1)$, there is no more busy-time budget for opening a new machine to schedule a job of length $T - \omega + 3$ at $[\omega - 2, T + 1)$. Furthermore, ALG cannot schedule any of these jobs on the machine with a job of length $T - \omega + 3$. Otherwise, the total busy-time will be $T + 1$ which is greater than the given time budget T. Hence, ALG can only gain throughput $tput^A \leq 1$. However, an optimal solution will schedule only jobs of length $T - \omega + 3$ and drop the first job of length $\omega - 1$. We need to consider the case when $T - \omega + 3 < \omega$ and $T - \omega + 3 \geq \omega$. Since $T = \omega$, $T - \omega + 3 = 3$, and thus, these two cases equivalent to the case when $\omega > 3$ and $\omega \leq 3$.

In the case when $\omega > 3$, an optimal solution will gain throughput $tput^* \geq g\lfloor \frac{T}{T-\omega+3} \rfloor$. Since $\frac{T}{T-\omega+3} = \frac{T}{3}$, $tput^* \geq g\lfloor \frac{T}{T-\omega+3} \rfloor = g\lfloor \frac{T}{3} \rfloor$.

In the case when $\omega \leq 3$, an optimal solution will gain throughput $tput^* \geq 2g\lfloor \frac{T}{T-\omega+3} \rfloor$. Since jobs of length $T - \omega + 3$ are feasible, $T \geq T - \omega + 3 = 3$. Furthermore, $T = \omega = 3$ because $\omega \leq 3$. Thus, an optimal solution will gain throughput $tput^* \geq 2g\lfloor \frac{T}{T-\omega+3} \rfloor = \lfloor \frac{T}{3} \rfloor = 2g$.

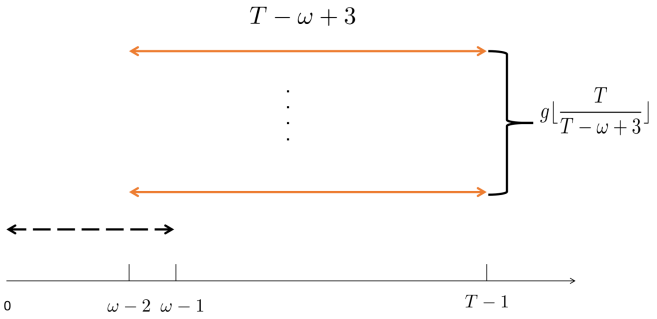

Figure 3: Theorem 9, case 3 (dotted lines are online schedule, solid lines are optimal schedule)

As a result, the competitive ratio is at least $2g$ when $T = \omega = 3$, or $g\lfloor \frac{T}{3} \rfloor$ when $T = \omega > 3$, corresponding to lower bound (3) and (4).
Hence, **Theorem 9** is proven.

■

Theorem 10 *For infeasible clique instances of* ***Categorized Weighted Throughput Maximization Problem****, when the* `greedy algorithm` *gains throughput at least 4, $T \geq 4$ and $T > \omega$, it is an*

optimal deterministic online algorithm.

Proof. According to **Lemma 2**, we know an optimal solution can gain throughput at most gT. We discuss 3 cases: when the greedy algorithm gains throughput $tput^A \geq 4$, $4 > tput^A \geq 2$ and $tput^A = 1$.
Case 1: When the `greedy algorithm` gains throughput $tput^A \geq 4$, the competitive ratio is at most $\frac{gT}{4}$ which is bounded by the lower bound of the competitive ratio when $T \geq 4$ and $T > \omega$, which is proven in **Theorem 9**.

Hence, **Theorem 10** is proven.

■

5.1.3 Infeasible one-sided clique instances

Theorem 11 *For infeasible one-sided clique instances of* ***Categorized Weighted Throughput Maximization Problem****, no deterministic online algorithm has a competitive ratio lower than* $\frac{gT}{g+1}$ *when* $T \geq \frac{2}{g} + 2$*, or lower than 2 when* $2 \leq T < \frac{2}{g} + 2$

Proof. Consider the following adversary:

Case 1: When $T \geq \frac{2}{g} + 2$, the adversary first releases a job of length T at $[0, T)$. Any online algorithm ALG has to schedule this job according to **Proposition 1** and consume T units of busy-time budget. Then, the adversary releases gT jobs of length 1 at $[0, 1)$. Since ALG already consumed T units of busy-time budget when scheduling a job of length T at $[0, T)$, therefore there is no more busy-time budget for opening a new machine to schedule a job of length 1 at $[0, 1)$. Hence, ALG can only schedule at most $g - 1$ of these jobs on the machine with a job of length T, and thus $tput^A \leq g + 1$. However, an optimal solution will schedule only these jobs and drop the first job of length T and $tput^* \geq gT$. As a result, the competitive ratio is at least $\frac{gT}{g+1}$.

Case 2: When $2 \leq T < \frac{2}{g} + 2$, the adversary keeps releasing jobs of 1 until one machine is used and filled with g jobs, or two machines are used. We consider two sub-cases:

Case 2.1: one machine is used by the online algorithm ALG with at most $g - 1$ jobs of 1 being scheduled at $[0, 1)$. Hence, ALG can gain throughput $tput^A \leq g - 1$ while an optimal solution will schedule gT jobs of 1 at $[0, 1)$ and gain throughput $tput^* \geq gT$. Thus, the competitive ratio is at least $\frac{gT}{g-1}$.

Case 2.2: one machine is used by the online algorithm ALG with g jobs of 1 being scheduled at $[0, 1)$ or two machines are used by ALG. Then, the adversary releases g jobs of length T at $[0, T)$. Since ALG already consumed 1 units of busy-time budget when scheduling a job of length 1 at $[0, 1)$, therefore the remaining $T - 1$ units of busy-time budget is not enough to schedule a job of length T at $[0, T)$. Hence, ALG cannot schedule any of these jobs, and thus $tput^A \leq g$. However, an optimal solution will schedule only these jobs and drop the first job of length 1 and $tput^* \geq 2g$. As a result, the competitive ratio is at least 2.

Since $\frac{gT}{g-1} > T \geq 2$, the lower bound is 2 for **Case 2**.

The competitive ratio for **Case 1** is at least $\frac{gT}{g+1}$, and the competitive ratio for **Case 2** is at least 2. Furthermore, $\frac{gT}{g+1}$ is a tighter lower bound than 2 when $T \geq \frac{2}{g} + 2$. Hence, **Theorem 11** is proven.

■

Theorem 12 *For infeasible one-sided clique instances of* ***Categorized Weighted Throughput Maximization Problem****, the* `greedy algorithm` *is an optimal deterministic online algorithm if the following condition is satisfied:*
(1) $tput^A < g$.
(2) $tput^A \geq g + 1$ *and* $T > 2$.

Proof. According to **Lemma 2**, an optimal solution can gain throughput at most gT. We consider the two cases mentioned above.

Case 1: When the `greedy algorithm` gains throughput $tput^A < g$, it must have scheduled fewer than g jobs, otherwise, $tput^A \geq g$. Since this is a one-sided clique instance and all released jobs have length at most T, each released job could be scheduled on a single processor. Moreover, there are g processors on one machine so g jobs can be processed simultaneously on one machine.

Thus, the `greedy algorithm` must be able to schedule at least the first g jobs if there are more than g jobs released. Furthermore, when the `greedy algorithm` gains throughput $tput^A < g$, there is less than g jobs released and the `greedy algorithm` will schedule the same number of jobs as an optimal solution does, which means the competitive ratio is 1 in this case.

Case 2: When the `greedy algorithm` gains throughput $tput^A \geq g+1$, the competitive ratio is lower than or equal to $\frac{gT}{g+1}$ which can be bounded by the lower bound of the competitive ratio when $T > 2$.

Hence, **Theorem 12** is proven.

■

5.2 Feasible instances

In this section, we consider feasible instances. In infeasible instances, the adversary can trap the online algorithm by releasing some very long or very short job as the first job, while an optimal solution can avoid this trap by not scheduling this job. However, in feasible instances, there exists a schedule which can schedule all the released jobs, which makes it much harder for the adversary to trap the online algorithm. Therefore, the lower bound of the competitive ratio is improved significantly.

5.2.1 Feasible one-sided clique instances

Theorem 13 *For feasible one-sided clique instances of* ***Categorized Weighted Throughput Maximization Problem****, no deterministic online algorithm has a competitive ratio lower than* $\frac{4}{3}$*, when* $T < 2\omega - 1$*, or* $\frac{9}{7}$*, when* $T \geq 2\omega - 1$*.*

Proof. Consider the following adversary:
The adversary first releases a job of length $T+1-\omega$ at $[0, T+1-\omega)$. Any online algorithm ALG has to schedule this job according to **Proposition 1** and consume $T+1-\omega$ units of busy-time budget. Then, the adversary keeps releasing jobs of length 1 at $[0, 1)$ until one of the following conditions is satisfied:
(1) If x is the number of released jobs of length 1 and x^A is the number of jobs of length 1 scheduled by ALG. The condition is $x \geq \frac{3x^A+1}{2}$, when $T+1-\omega < \omega$, or $x \geq \frac{3x^A+2}{2}$, when $T+1-\omega \geq \omega$. This condition happens when ALG rejects too many jobs.
(2) One machine is full.
(3) One new machine is used by ALG.

Noted that when condition (2) or (3) happens, condition (1) cannot happen. Because condition (1) is triggered when ALG rejects a job and rejecting a job cannot trigger condition (2) or (3). Moreover, condition (2) or (3) happens when ALG accepts a job and once either of them is triggered, the adversary stops releasing jobs of length 1. Thus, after triggering condition (2) or (3), there is no more jobs of length 1 could be rejected.

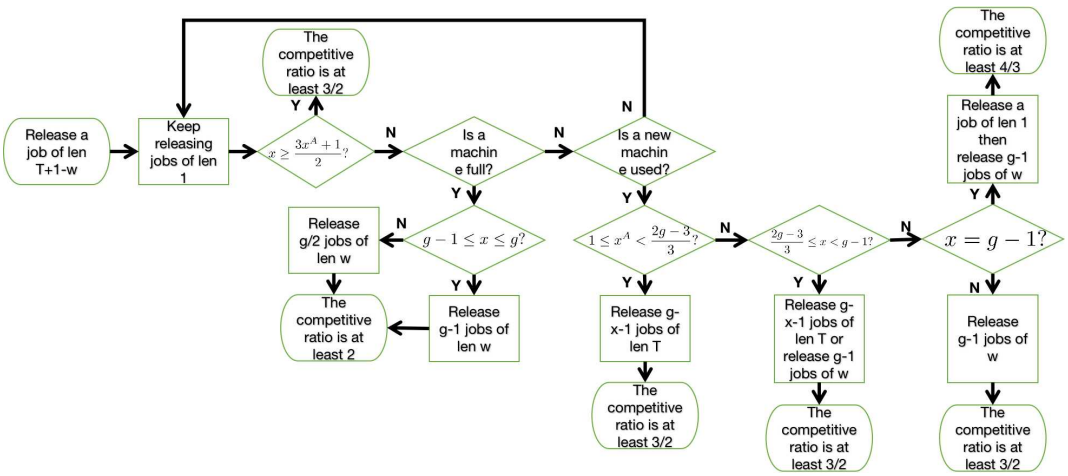

Figure 4: The process graph of Theorem 13(when $T + 1 - \omega < \omega$)

Case 1: When $T + 1 - \omega < \omega$, $x \geq \frac{3x^A+1}{2}$, we prove that the competitive ratio is at least $\frac{3}{2}$. In this case, an optimal solution schedules the first job of length $T + 1 - \omega$ and x jobs of length 1. Since $x \geq \frac{3x^A+1}{2}$, an optimal obtain throughput $tput^* \geq \frac{3x^A+3}{2}$. While ALG schedules the first job of length $T + 1 - \omega$ and only x^A jobs of length 1. Thus, ALG obtains throughput $tput^A \leq x^A + 1$. Hence the competitive ratio is at least $\frac{3x^A+3}{2x^A+2} = \frac{3}{2}$. The similar analysis can be applied to the case when $T + 1 - \omega \geq \omega$, $x \geq \frac{3x^A+2}{2}$.

Case 2: One machine is full. We prove the competitive ratio is at least $\frac{3}{2}$. In this case, ALG schedules the first job of length $T + 1 - \omega$ and $g - 1$ jobs of length 1 on the same machine. Thus, $x^A = g - 1$. There are two cases that need to be discussed, which are when $T + 1 - \omega < \omega$, and when $T + 1 - \omega \geq \omega$.

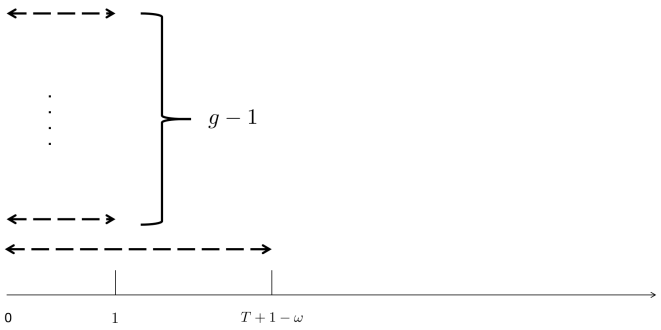

Figure 5: Theorem 13, case 2, online algorithm schedule

Case 2.1: when $T + 1 - \omega < \omega$. In this case, $x < \frac{3x^A+1}{2}$. Thus, $g - 1 \leq x < \frac{3g-2}{2}$. There are two cases that need to be discussed here which are when $g - 1 \leq x \leq g$ and when $g + 1 \leq x < \frac{3g-2}{2}$.

Case 2.1.1: when $g - 1 \leq x \leq g$. In this case, the adversary releases $g - 1$ jobs of length ω. Since ALG already consumed $T + 1 - \omega$ units of busy-time budget when scheduling a job of length $T + 1 - \omega$ and $g - 1$ jobs of length 1, and the machine is full, there is no busy-time budget left for opening a new machine to schedule a job of length ω. Hence, ALG cannot schedule any of these jobs, and thus, $tput^A \leq g$. However, an optimal solution will schedule the first job of length $T+1-\omega$ and $g - 1$ jobs of length ω on the same machine, and schedule all released jobs of length 1 on a new machine. Therefore, the optimal solution obtains throughput $tput^* \geq 2(g - 1) + x + 1 = 2g + x - 1$. Since $x \geq g - 1$, $tput^* \geq 2g + x - 1 \geq 3g - 2$. As a result, the competitive ratio is at least $\frac{3g-2}{g} = 3 - \frac{2}{g} \geq 2$ because $g \geq 2$.

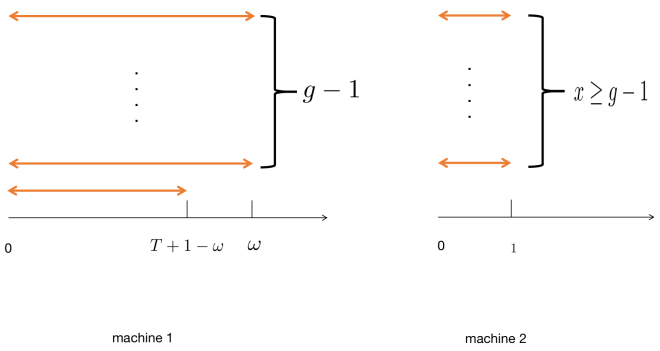

Figure 6: Theorem 13, case 2.1.1, optimal schedule

Case 2.1.2: when $g+1 \leq x < \frac{3g-2}{2}$. In this case, the adversary releases $\frac{g}{2}$ jobs of length ω. Since ALG already consumed $T+1-\omega$ units of busy-time budget when scheduling a job of length $T+1-\omega$ and $g-1$ jobs of length 1, and the machine is full, there is no busy-time budget left for opening a new machine to schedule a job of length ω. Hence, ALG cannot schedule any of these jobs, and thus, $tput^A \leq g$. However, an optimal solution will schedule g jobs of length 1 on one machine, and open a new machine to schedule the remaining $x-g$ jobs of length 1, the first job of length $T+1-\omega$ and $\frac{g}{2}$ jobs of length ω. Therefore, the optimal solution obtains throughput $tput^* \geq g+1+x$. Since $x \geq g+1$, $tput^* \geq 2g+2$. As a result, the competitive ratio is at least $\frac{2g+2}{g} > 2$.

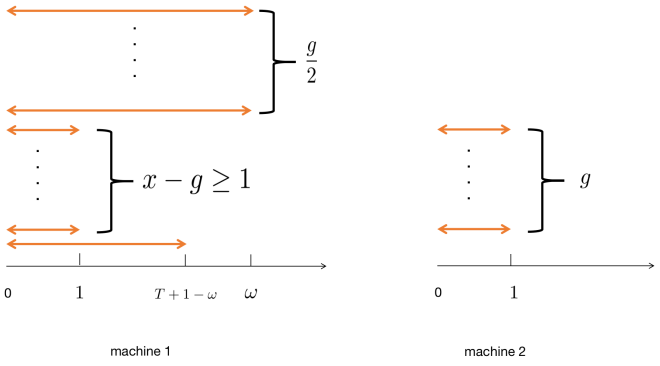

Figure 7: Theorem 13, case 2.1.2, optimal schedule

Case 2.2: when $T+1-\omega \geq \omega$. In this case, $x < \frac{3x^A+2}{2}$. Thus, $g-1 \leq x < \frac{3g-1}{2}$. There are two cases that need to be discussed here which are when $g-1 \leq x \leq g$ and when $g+1 \leq x < \frac{3g-1}{2}$.

Case 2.2.1: when $g-1 \leq x \leq g$. In this case, the adversary releases $g-1$ jobs of length ω. Since ALG already consumed $T+1-\omega$ units of busy-time budget when scheduling a job of length $T+1-\omega$ and $g-1$ jobs of length 1, and the machine is full, there is no busy-time budget left for opening a new machine to schedule a job of length ω. Hence, ALG cannot schedule any of these jobs, and thus, $tput^A \leq g+1$. However, an optimal solution will schedule the first job of length $T+1-\omega$ and $g-1$ jobs of length ω on the same machine, and schedule all released jobs of length 1 on a new machine. Therefore, the optimal solution obtains throughput $tput^* \geq 2(g-1)+x+2 = 2g+x$. Since $x \geq g-1$, $tput^* \geq 2g+x \geq 3g-1$. As a result, the competitive ratio is at least $\frac{3g-1}{g+1} = 3-\frac{4}{g+1} \geq \frac{5}{3}$ because $g \geq 2$.

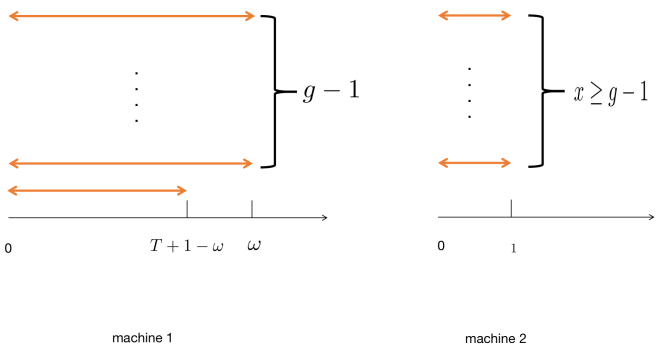

Figure 8: Theorem 13, case 2.2.1, optimal schedule

Case 2.2.2: when $g+1 \leq x < \frac{3g-1}{2}$. In this case, the adversary releases $\frac{g}{2}$ jobs of length ω. Since ALG already consumed $T+1-\omega$ units of busy-time budget when scheduling a job of length $T+1-\omega$ and $g-1$ jobs of length 1, and the machine is full, there is no busy-time budget left for opening a new machine to schedule a job of length ω. Hence, ALG cannot schedule any of these jobs, and thus, $tput^A \leq g+1$. However, an optimal solution will schedule g jobs of length 1 on one machine, and open a new machine to schedule the remaining $x-g$ jobs of length 1, the first job of length $T+1-\omega$ and $\frac{g}{2}$ jobs of length ω. Therefore, the optimal solution obtains throughput $tput^* \geq g+2+x$. Since $x \geq g+1$, $tput^* \geq 2g+3$. As a result, the competitive ratio is at least $\frac{2g+3}{g+1} > 2$.

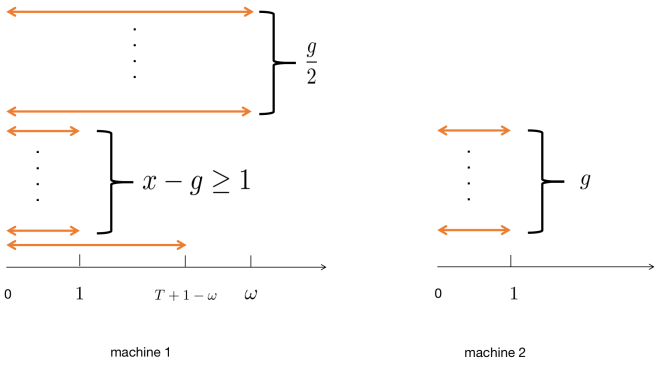

Figure 9: Theorem 13, case 2.2.2, optimal schedule

Case 3: One new machine is used by ALG. We prove that the competitive ratio is at least $\frac{4}{3}$ when $T+1-\omega < \omega$, or $\frac{9}{7}$ when $T+1-\omega \geq \omega$. In this case, ALG schedules the first job of length $T+1-\omega$ on one machine. and one job of length 1 on a new machine. However, there is no full machine, and adversary stops releasing jobs of length 1 once a new machine is used. Thus, there are at most $g-2$ jobs of length 1 scheduled by ALG on the machine with the job of length $T+1-\omega$. Otherwise, there must be a full machine. Thus, summing up the only job of length 1 on the new machine, $1 \leq x^A \leq g-1$. There are two cases that need to be discussed, which are when $T+1-\omega < \omega$, and when $T+1-\omega \geq \omega$.

Case 3.1: when $T+1-\omega < \omega$. In this case, $x < \frac{3x^A+1}{2}$. Thus, $g-1 \leq x < \frac{3g-2}{2}$. There are two cases that need to be discussed here which are when $1 \leq x^A < \frac{2g-3}{3}$ and when $\frac{2g-3}{3} \leq x^A \leq g-1$.

Case 3.1.1: when $1 \leq x^A < \frac{2g-3}{3}$. In this case, since $x < \frac{3x^A+1}{2}$, $1 \leq x < g-1$. The adversary releases $g-x-1$ jobs of length T. Since ALG already scheduled a job of length $T+1-\omega$ on one machine, and a job of length 1 on a new machine, there is no busy-time budget left for scheduling any job of length T on the existing machine or opening a new machine to schedule a job of length T. Hence, ALG cannot schedule any of these jobs. Thus, $tput^A \leq x^A+1$. Furthermore, since

$1 \leq x^A < \frac{2g-3}{3}$, $tput^A \leq x^A+1 < \frac{2g}{3}$. However, an optimal solution will schedule all the released jobs on the same machine, which are the first job of length $T+1-\omega$, x jobs of length 1, and $g-x-1$ jobs of length T. Therefore, the optimal solution obtains throughput $tput^* \geq 2(g-x-1)+x+1 = 2g-x-1$. Hence, the competitive ratio is at least $\frac{3(2g-x-1)}{2g} = \frac{6g-3x-3}{2g} = 3 - \frac{3x+3}{2g}$. Furthermore, since $x < g-1$, $3 - \frac{3x+3}{2g} > 3 - \frac{3(g-1)+3}{2g} = 3 - \frac{3g}{2g} = \frac{3}{2}$. As a result the competitive ratio is at least $\frac{3}{2}$.

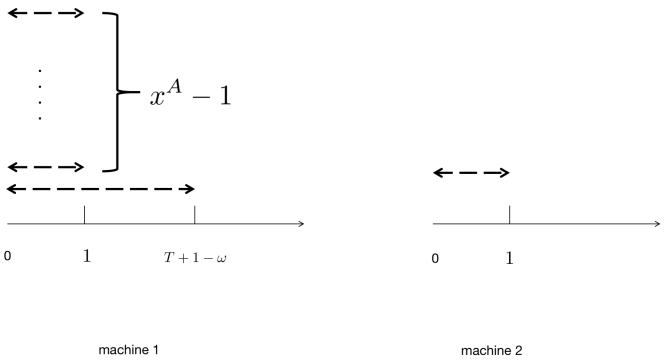

Figure 10: Theorem 13, case 3.1.1, online algorithm schedule

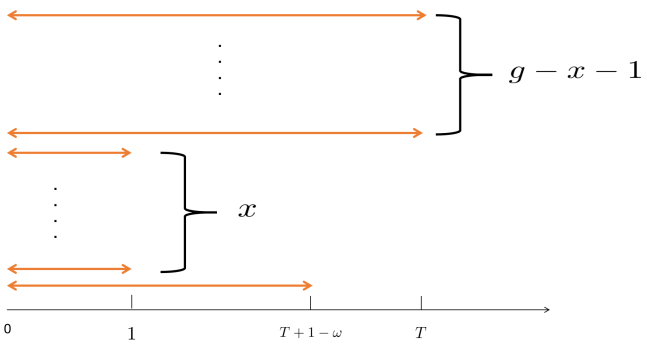

Figure 11: Theorem 13, case 3.1.1, optimal schedule

Case 3.1.2: when $\frac{2g-3}{3} \leq x^A \leq g-1$. In this case, since $x < \frac{3x^A+1}{2}$, $\frac{2g-3}{3} \leq x < \frac{3g-2}{2}$. There are three cases that need to be discussed here which are when $\frac{2g-3}{3} \leq x < g-1$, when $x = g-1$, and when $g \leq x < \frac{3g-2}{2}$.

Case 3.1.2.1: $\frac{2g-3}{3} \leq x < g-1$. In this case, the adversary will release two different jobs depending on the value of g, x, and x^A. And the adversary will release the type of job that leads to a higher competitive ratio. We prove that in this case, the competitive ratio is at least $\frac{3}{2}$.

The adversary releases $g-x-1$ jobs of length T. Since ALG already scheduled a job of length $T+1-\omega$ on one machine, and a job of length 1 on a new machine, there is no busy-time budget left for scheduling any job of length T on the existing machine or opening a new machine to schedule a job of length T. Hence, ALG cannot schedule any of these jobs. Thus, $tput^A \leq x^A+1$. However, an optimal solution will schedule all the released jobs on the same machine, which are the first job of length $T+1-\omega$, x jobs of length 1, and $g-x-1$ jobs of length T. Therefore, the optimal solution obtains throughput $tput^* \geq 2(g-x-1)+x+1 = 2g-x-1$. Hence, the competitive ratio is at least $\frac{2g-x-1}{x^A+1}$.

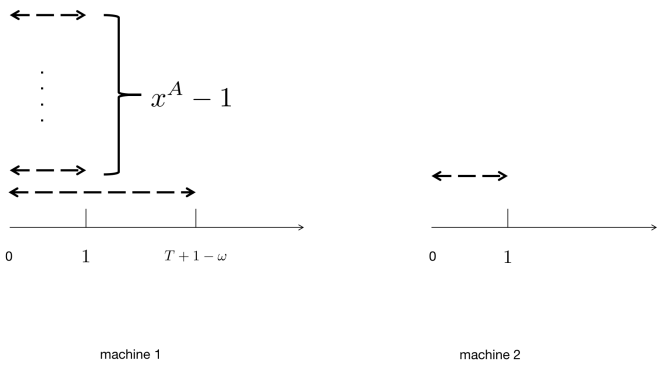

Figure 12: Theorem 13, case 3.1.2.1, online algorithm schedule (job T)

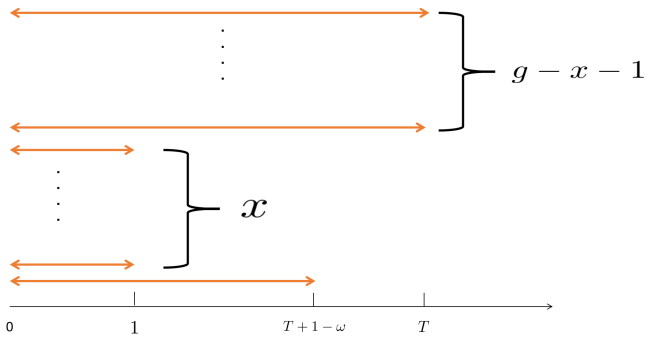

Figure 13: Theorem 13, case 3.1.2.1, optimal schedule (job T)

The adversary releases $g-1$ jobs of length ω. Since ALG already scheduled a job of length $T+1-\omega$ on one machine, and a job of length 1 on a new machine, ALG can only schedule these jobs of length ω on the machine with jobs of length $T+1-\omega$. Otherwise, if ALG schedules any job of length ω on the machine with only one job of length 1, or open a new machine to schedule jobs of length ω, it will use more than T busy-time. Thus, $tput^A \leq 2(g-x^A)+x^A+1 = 2g-x^A+1$. However, an optimal solution will schedule all the jobs of length 1 on the same machine, and schedule the jobs of length ω with the job of length of $T+1-\omega$ on the same machine. Therefore, the optimal solution obtains throughput $tput^* \geq 2(g-1)+x+1 = 2g+x-1$. Hence, the competitive ratio is at least $\frac{2g+x-1}{2g-x^A+1}$.

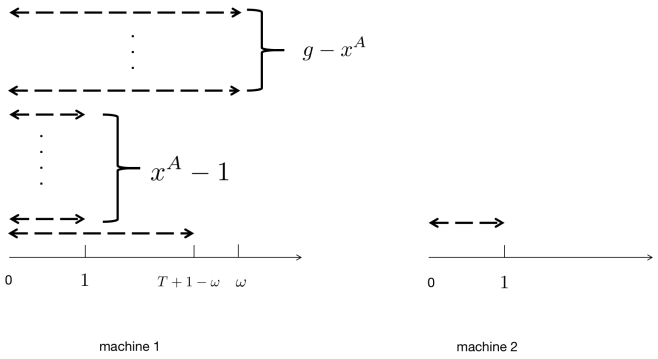

Figure 14: Theorem 13, case 3.1.2.1, online algorithm schedule (job w)

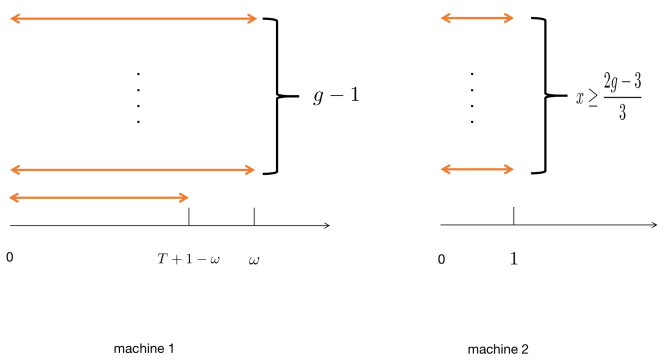

Figure 15: Theorem 13, case 3.1.2.1, optimal schedule (job w)

By running experiments, it is found that the lowest competitive ratio is obtained when $g = 5, x = 3, x^A = 3$, and the competitive ratio is exactly $\frac{3}{2}$. This is one of the lowest intersection points of the two competitive ratio functions. This also makes sense when looking at the graph of the two functions because the values given by one of the competitive ratio function decreases with x and x^A increasing, while the other ones increase with x and x^A increasing. As a result the competitive ratio of **Case 3.1.2.1** is at least $\frac{3}{2}$.

Case 3.1.2.2: $x = g - 1$. In this case, the adversary will release one more job of length 1. Thus, $x = g$. Furthermore, since $x < \frac{3x^A+1}{2}$, and ALG might schedule the last job of length 1, $\frac{2g-1}{3} < x^A \leq g$.

The adversary releases $g - 1$ jobs of length ω. Since ALG already scheduled a job of length $T + 1 - \omega$ on one machine, and a job of length 1 on a new machine, ALG can only schedule these jobs of length ω on the machine with jobs of length $T + 1 - \omega$. Otherwise, if ALG schedules any job of length ω on the machine with only one job of length 1, or open a new machine to schedule jobs of length ω, it will use more than T busy-time. Moreover, the last released job of length 1 could be scheduled by ALG. Thus, $tput^A \leq 2(g - x^A) + x^A + 2 = 2g - x^A + 2$. However, an optimal solution will schedule the jobs of length ω with the job of length of $T + 1 - \omega$ on one machine. And it will schedule all g jobs of length 1 on a second machine. Therefore, the optimal solution obtains throughput $tput^* \geq 2(g - 1) + g + 1 = 3g - 1$. Hence, the competitive ratio is at least $\frac{3g-1}{2g-x^A+2}$. By running experiments, it is found that the lowest competitive ratio is obtained when $g = 3, x = 3, x^A = 2$, and the competitive ratio is exactly $\frac{4}{3}$. Thus, the competitive ratio is at least $\frac{4}{3}$.

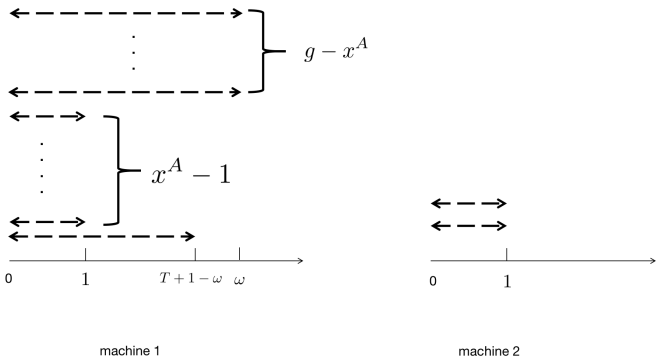

Figure 16: Theorem 13, case 3.1.2.2, online algorithm schedule

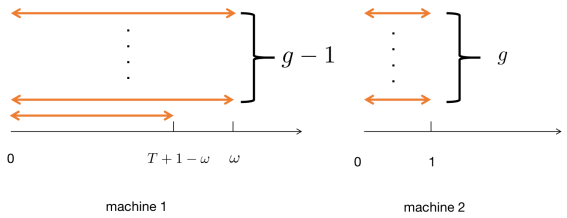

Figure 17: Theorem 13, case 3.1.2.2, optimal schedule

Case 3.1.2.3: $g \leq x < \frac{3g-2}{2}$. In this case, since $x < \frac{3x^A+1}{2}$, $\frac{2g-1}{3} < x^A \leq g-1$. The adversary releases $g-1$ jobs of length ω. Since ALG already scheduled a job of length $T+1-\omega$ on one machine, and a job of length 1 on a new machine, ALG can only schedule these jobs of length ω on the machine with jobs of length $T+1-\omega$. Otherwise, if ALG schedules any job of length ω on the machine with only one job of length 1, or open a new machine to schedule jobs of length ω, it will use more than T busy-time. Thus, $tput^A \leq 2(g-x^A)+x^A+1 = 2g-x^A+1$. Since $\frac{2g-1}{3} < x^A$, $tput^A \leq 2g-x^A+1 < \frac{4g+4}{3}$. However, an optimal solution will schedule the jobs of length ω with the job of length of $T+1-\omega$ on one machine. And it will schedule at least g jobs of length 1 on a second machine since $x \geq g$. If there is any job of length 1 left after scheduling g of them, then it will schedule the remaining jobs of length 1 on a third machine. Therefore, the optimal solution obtains throughput $tput^* \geq 2(g-1)+1+x$. Since $x \geq g$, $tput^* \geq 2(g-1)+1+x \geq 2(g-1)+1+g = 3g-1$. Hence, the competitive ratio is at least $\frac{3g-1}{2g-x^A+1} > \frac{9g-3}{4g+4}$. Since $g \geq 3$, $\frac{9g-3}{4g+4} \geq \frac{3}{2}$. Thus, the competitive ratio is at least $\frac{3}{2}$.

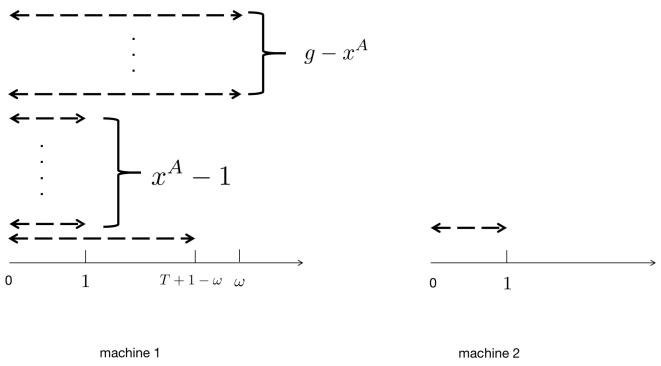

Figure 18: Theorem 13, case 3.1.2.3, online algorithm schedule

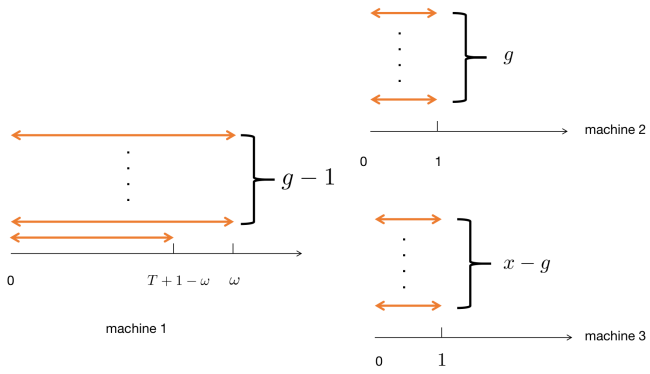

Figure 19: Theorem 13, case 3.1.2.3, optimal schedule

Case 3.2: when $T+1-\omega \geq \omega$. In this case, $x < \frac{3x^A+2}{2}$. Thus, $g-1 \leq x < \frac{3g-1}{2}$. There are two cases that need to be discussed here which are when $1 \leq x^A < \frac{2g-4}{3}$ and when $\frac{2g-4}{3} \leq x^A \leq g-1$.

Case 3.2.1: when $1 \leq x^A < \frac{2g-4}{3}$. In this case, since $x < \frac{3x^A+2}{2}$, $1 \leq x < g-1$. The adversary releases $g-x-1$ jobs of length T. Since ALG already scheduled a job of length $T+1-\omega$ on one machine, and a job of length 1 on a new machine, there is no busy-time budget left for scheduling any job of length T on the existing machine or opening a new machine to schedule a job of length T. Hence, ALG cannot schedule any of these jobs. Thus, $tput^A \leq x^A + 2$. Furthermore, since $x^A < \frac{2g-4}{3}$, $tput^A \leq x^A + 2 < \frac{2g+2}{3}$. However, an optimal solution will schedule all the released jobs on the same machine, which are the first job of length $T+1-\omega$, x jobs of length 1, and $g-x-1$ jobs of length T. Therefore, the optimal solution obtains throughput $tput^* \geq 2(g-x-1)+x+2 = 2g-x$. Hence, the competitive ratio is at least $\frac{3(2g-x)}{2g+2} = \frac{6g-3x}{2g+2}$. Furthermore, since $x < g-1$, $\frac{6g-3x}{2g+2} > \frac{6g-3(g-1)}{2g+2} = \frac{3g+3}{2g+2} = \frac{3}{2}$. As a result the competitive ratio is at least $\frac{3}{2}$.

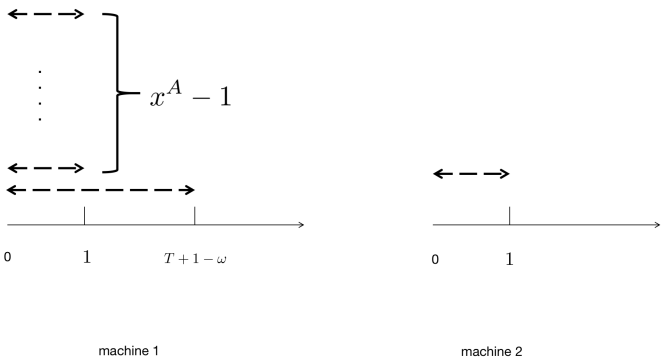

Figure 20: Theorem 13, case 3.2.1, online algorithm schedule

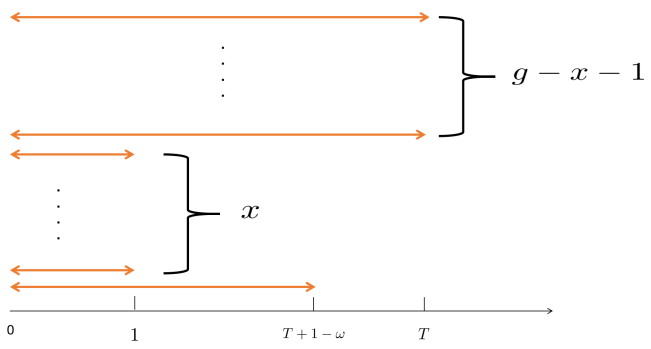

Figure 21: Theorem 13, case 3.2.1, optimal schedule

Case 3.2.2: when $\frac{2g-4}{3} \leq x^A \leq g-1$. In this case, since $x < \frac{3x^A+2}{2}$, $\frac{2g-4}{3} \leq x < \frac{3g-1}{2}$. There are three cases that need to be discussed here which are when $\frac{2g-4}{3} \leq x < g-1$, when $x = g-1$, and when $g \leq x < \frac{3g-1}{2}$.

Case 3.2.2.1: $\frac{2g-4}{3} \leq x < g-1$. In this case, the adversary will release two different jobs depending on the value of g, x, and x^A. And the adversary will release the type of job that leads to a higher competitive ratio. We prove that in this case, the competitive ratio is at least $\frac{3}{2}$.

The adversary releases $g-x-1$ jobs of length T. Since ALG already scheduled a job of length $T+1-\omega$ on one machine, and a job of length 1 on a new machine, there is no busy-time budget left for scheduling any job of length T on the existing machine or opening a new machine to schedule a job of length T. Hence, ALG cannot schedule any of these jobs. Thus, $tput^A \leq x^A+2$. However, an optimal solution will schedule all the released jobs on the same machine, which are the first job of length $T+1-\omega$, x jobs of length 1, and $g-x-1$ jobs of length T. Therefore, the optimal solution obtains throughput $tput^* \geq 2(g-x-1)+x+2 = 2g-x$. Hence, the competitive ratio is at least $\frac{2g-x}{x^A+2}$.

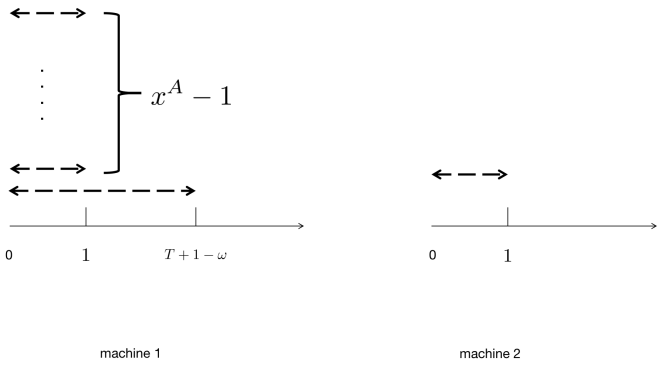

Figure 22: Theorem 13, case 3.2.2.1, online algorithm schedule (job T)

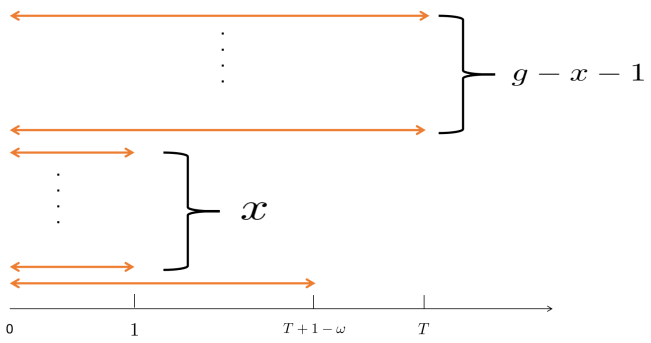

Figure 23: Theorem 13, case 3.2.2.1, optimal schedule (job T)

The adversary releases $g-1$ jobs of length ω. Since ALG already scheduled a job of length $T+1-\omega$ on one machine, and a job of length 1 on a new machine, ALG can only schedule these jobs of length ω on the machine with jobs of length $T+1-\omega$. Otherwise, if ALG schedules any job of length ω on the machine with only one job of length 1, or open a new machine to schedule jobs of length ω, it will use more than T busy-time. Thus, $tput^A \leq 2(g-x^A)+x^A+2 = 2g-x^A+2$. However, an optimal solution will schedule all the jobs of length 1 on the same machine, and schedule the jobs of length ω with the job of length of $T+1-\omega$ on the same machine. Therefore, the optimal solution obtains throughput $tput^* \geq 2(g-1)+x+2 = 2g+x$. Hence, the competitive ratio is at least $\frac{2g+x}{2g-x^A+2}$.

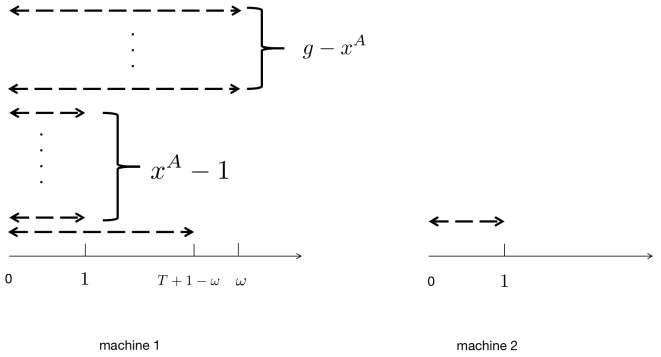

Figure 24: Theorem 13, case 3.2.2.1, online algorithm schedule (job w)

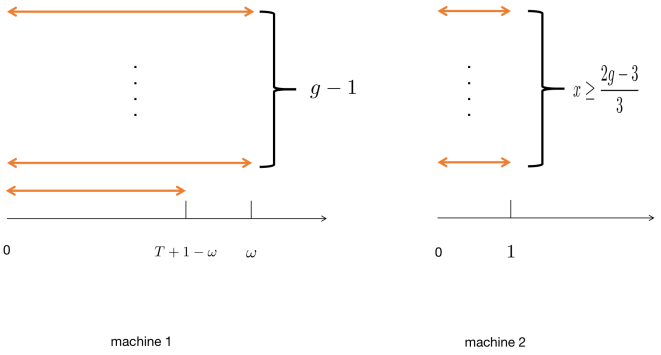

Figure 25: Theorem 13, case 3.2.2.1, optimal schedule (job w)

By running experiments, it is found that the lowest competitive ratio is obtained when $g = 4, x = 2, x^A = 2$, and the competitive ratio is exactly $\frac{3}{2}$. This is one of the lowest intersection points of the two competitive ratio functions. This also makes sense when looking at the graph of the two functions because the values given by one of the competitive ratio function decreases with x and x^A increasing, while the other ones increase with x and x^A increasing. As a result, the competitive ratio of **Case 3.2.2.1** is at least $\frac{3}{2}$.

Case 3.2.2.2: $x = g - 1$. In this case, the adversary will release one more job of length 1. Thus, $x = g$. Furthermore, since $x < \frac{3x^A+2}{2}$, and ALG might schedule the last job of length 1, $\frac{2g-2}{3} < x^A \leq g$.

The adversary releases $g - 1$ jobs of length ω. Since ALG already scheduled a job of length $T + 1 - \omega$ on one machine, and a job of length 1 on a new machine, ALG can only schedule these jobs of length ω on the machine with jobs of length $T + 1 - \omega$. Otherwise, if ALG schedules any job of length ω on the machine with only one job of length 1, or open a new machine to schedule jobs of length ω, it will use more than T busy-time. Moreover, the last released job of length 1 could be scheduled by ALG. Thus, $tput^A \leq 2(g - x^A) + x^A + 3 = 2g - x^A + 3$. However, an optimal solution will schedule the jobs of length ω with the job of length of $T + 1 - \omega$ on one machine. And it will schedule all g jobs of length 1 on a second machine. Therefore, the optimal solution obtains throughput $tput^* \geq 2(g - 1) + x + 2 = 2g + x$. Hence, the competitive ratio is at least $\frac{2g+x}{2g-x^A+3}$. By running experiments, it is found that the lowest competitive ratio is obtained when $g = 3, x = 3, x^A = 2$, and the competitive ratio is exactly $\frac{9}{7}$. Thus, the competitive ratio is at least $\frac{9}{7}$.

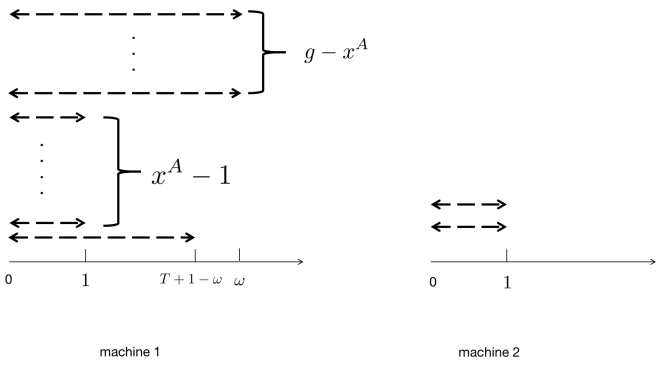

Figure 26: Theorem 13, case 3.2.2.2, online algorithm schedule

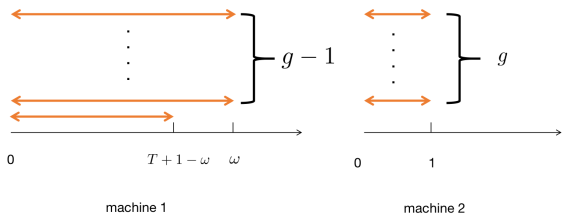

Figure 27: Theorem 13, case 3.2.2.2, optimal schedule

Case 3.2.2.3: $g \leq x < \frac{3g-1}{2}$. In this case, since $x < \frac{3x^A+2}{2}$, $\frac{2g-2}{3} < x^A \leq g - 1$. The adversary releases $g - 1$ jobs of length ω. Since ALG already scheduled a job of length $T + 1 - \omega$ on one

machine, and a job of length 1 on a new machine, ALG can only schedule these jobs of length ω on the machine with jobs of length $T+1-\omega$. Otherwise, if ALG schedules any job of length ω on the machine with only one job of length 1, or open a new machine to schedule jobs of length ω, it will use more than T busy-time. Thus, $tput^A \leq 2(g-x^A)+x^A+2 = 2g-x^A+2$. However, an optimal solution will schedule the jobs of length ω with the job of length of $T+1-\omega$ on one machine. And it will schedule at least g jobs of length 1 on a second machine since $x \geq g$. If there is any job of length 1 left after scheduling g of them, then it will schedule the remaining jobs of length 1 on a third machine. Therefore, the optimal solution obtains throughput $tput^* \geq 2(g-1)+2+x = 2g+x$. Hence, the competitive ratio is at least $\frac{2g+x}{2g-x^A+2}$. By running experiments, it is found that the lowest competitive ratio is obtained when $g = 3, x = 3, x^A = 2$, and the competitive ratio is exactly $\frac{3}{2}$. Thus, the competitive ratio is at least $\frac{3}{2}$.

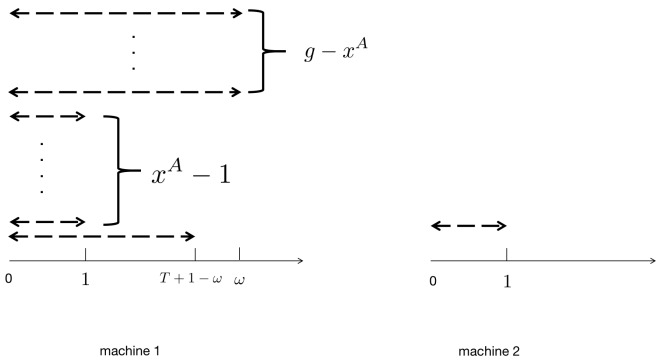

Figure 28: Theorem 13, case 3.2.2.3, online algorithm schedule

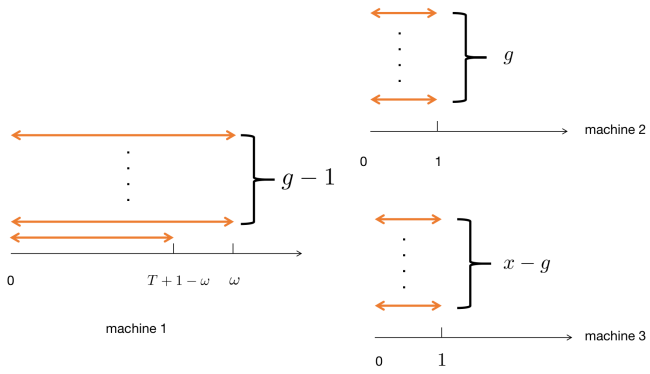

Figure 29: Theorem 13, case 3.2.2.3, optimal schedule

■

6 Resource augmentation

In this section, we will discuss infeasible instances of **Categorized Weighted Throughput Maximization Problem** with resource augmentation where the online algorithm is granted with extra resource such as extra processors on machines or extra busy-time budgets.

The objective is to maximize the obtained throughput by scheduling jobs using the augmented resources.

6.1 Infeasible general instances

Theorem 14 *For infeasible general instances of* ***Categorized Weighted Throughput Maximization Problem*** *with extra T busy-time budget for the online algorithm, no deterministic online algorithm has a competitive ratio lower than g when $2 \leq T \leq 4$, or $\frac{gT}{4}$ when $T > 4$.*

Proof. Consider the following adversary:

The adversary first releases a job of length T at $[0, T)$. Any online algorithm ALG has to schedule this job according to **Proposition 1** and consume T units of busy-time budget. Then, the adversary keeps releasing jobs of length T at $[T, 2T)$ until one job of length T is scheduled by ALG at $[T, 2T)$ or g jobs of length T are released at $[T, 2T)$.

Case 1: one job of length T is scheduled by *ALG* at [T,2T). In this case, the adversary releases gT jobs of length 1 at $[2T, 2T + 1)$. Since ALG already consumed $2T$ units of busy-time budget when scheduling a job of length T at $[0, T)$ and a job of length T at $[T, 2T)$, therefore there is no busy-time budget left for opening a new machine to schedule a job of length 1 at $[2T, 2T + 1)$. Furthermore, ALG cannot schedule any jobs of length 1 at $[2T, 2T + 1)$ on any machine, otherwise, ALG will use $2T+1$ busy-time which is more than the given busy-time budget $2T$. Thus, $tput^A \leq 4$. However, an optimal solution can schedule all the jobs of length 1 at $[2T, 2T + 1)$, and drop the jobs of length T, and thus, an optimal solution will gain throughput $tput^* \geq gT$. As a result, the competitive ratio is at least $\frac{gT}{4}$.

Case 2: g jobs of length T are released at [T,2T). In this case, since ALG scheduled no job of length T at $[T, 2T)$, ALG will gain throughput $tput^A \leq 2$. While an optimal solution drops the first job of length T released at $[0, T)$ and schedule all jobs of length T at $[T, 2T)$. Therefore, an optimal solution will gain throughput $tput^* \geq 2g$. As a result, the competitive ratio is at least g.

Hence, **Theorem 14** is proven.

■

Theorem 15 *For infeasible general instances of* ***Categorized Weighted Throughput Maximization Problem*** *with extra g processors for the online algorithm, no deterministic online algorithm has a competitive ratio lower than $\frac{gT}{2}$ when $T \geq 4$, or $2g$ when $2 \leq T < 4$.*

Proof. Consider the same adversary in **Theorem 7**. Even if the online algorithm ALG is granted with extra g processors augmentation on every machine, it still cannot schedule more jobs than the case when it has only g processors on every machine. Hence, **Theorem 15** is proven.

■

6.2 Infeasible one-sided clique instances

Theorem 16 *For infeasible one-sided clique instances of* ***Categorized Weighted Throughput Maximization Problem*** *with extra g processors for the online algorithm, no deterministic online algorithm has a competitive ratio lower $\frac{gT}{2g-1}$ when $T \geq 3$.*

Proof. Consider the following adversary:

Case 1: when $T \geq 3$. In this case, the adversary first releases a job of length T at $[0, T)$. Any online algorithm ALG has to schedule this job according to **Proposition 1** and consume T units of busy-time budget. Then, the adversary releases gT jobs of length 1 at $[0, 1)$. Since ALG already consumed T units of busy-time budget when scheduling a job of length T at $[0, T)$, therefore there is no more busy-time budget for opening a new machine to schedule a job of length 1 at $[0, 1)$. Hence, ALG can only schedule at most $2g - 1$ of these jobs on the machine with a job of length T, and thus $tput^A \leq 2g + 1$. However, an optimal solution can schedule all the jobs but the first job of length T, and thus, $tput^* \geq gT$. As a result, the competitive ratio is at least $\frac{gT}{2g+1}$.

Hence, **Theorem 16** is proven.

■

Claim 1 *For infeasible one-sided clique instances of* ***Categorized Weighted Throughput Maximization Problem*** *with extra g processors for the online algorithm, if the* `greedy algorithm` *schedules fewer than $2g$ jobs, then, an optimal solution must schedule the same or fewer number of jobs.*

Proof. Since this is a one-sided clique instance and all released jobs have length at most T, any released job can be scheduled on a single processor. Moreover, for the `greedy algorithm`, there are $2g$ processors on one machine so at most $2g$ jobs can be processed simultaneously on one machine. Thus, if there are more than $2g$ jobs released, the `greedy algorithm` must be able to schedule at least the first $2g$ jobs. Furthermore, when the `greedy algorithm` schedules fewer than $2g$ jobs, there are less than $2g$ jobs released and the `greedy algorithm` will schedule the same number of jobs as an optimal solution does or even schedule more jobs than an optimal solution does since the optimal solution has only g processors on machines. Hence, **Claim 1** is proven.

■

Theorem 17 *For infeasible one-sided clique instances of* ***Categorized Weighted Throughput Maximization Problem*** *with extra g processors for the online algorithm, when the* `greedy algorithm` *gains throughput at least $2g+1$ it is an optimal deterministic online algorithm.*

Proof. According to **Lemma 2**, an optimal solution can gain throughput at most gT.

When the `greedy algorithm` gains throughput $tput^A \geq 2g+1$, the competitive ratio is lower than or equal to $\frac{gT}{2g+1}$. which can be bounded by the lower bound of the competitive ratio.
Hence, **Theorem 17** is proven.

■

7 Conclusion

Shalon ei al.[11] proposed and discussed the **Busy Time Weighted Throughput Maximization Problem** where any job has the same amount of weight, which might not correspond to the situation in the real world. So we proposed **Proportional Weighted Throughput Maximization Problem** and **Categorized Weighted Throughput Maximization Problem**.

For infeasible instances of **Proportional Weighted Throughput Maximization Problem**, we proved that no deterministic online algorithm has a constant competitive ratio by designing different adversaries. We also prove that the greedy algorithm is the optimal online algorithm. One possible further research direction is discussing this problem on feasible instances. In this case, the adversary cannot trap the online algorithm by releasing some extremely short jobs that an optimal solution does not schedule. Thus, it seems possible to obtain a lower competitive ratio, or even a constant competitive ratio as a negative result. Moreover, it might require the online algorithm to use a more flexible strategy to become an optimal algorithm.

For infeasible instances of **Categorized Weighted Throughput Maximization Problem**, we designed some adversaries to prove that no deterministic online algorithm has constant competitive ratio, unless it is a one-sided clique instance and T is 2. We also proved that the greedy algorithm is an optimal deterministic online algorithm if some conditions are satisfied. One possible further research direction is designing an optimal deterministic online algorithm for infeasible instances. In this paper, the competitive ratio of the greedy algorithm cannot always match the lower bound of the competitive ratio. However, the competitive ratio of a real optimal algorithm should always match the lower bound of the competitive ratio.

For feasible instances of **Categorized Weighted Throughput Maximization Problem**, we designed one complicated adversary for one-sided instances and proved no deterministic online algorithm has competitive ratio lower than $\frac{9}{7}$. There are many possible research problems for feasible instances. For example, designing adversaries for general or clique instances to figure out the lower bound of the competitive ratio, or designing a good online algorithm.

A possible research direction based on **Categorized Weighted Throughput Maximization Problem** is changing the value of the length threshold and the weight of jobs. For instance, in this paper, the weight of a job is always less than or equal to its length. But what if the weight of jobs could be greater than its length? Or what if the weight and the length of jobs are assigned arbitrarily?

For **resource augmentation**, we tried extra busy-time and extra processors on infeasible instances and some of them can help increase the lower bound of the competitive ratio. For example, both extra busy-time budgets and extra processors can help improve the lower bound competitive ratio of infeasible general instances. One possible further research direction is giving the online algorithm extra speed on machines. In this case, the released time and deadline of every job remain the same but it requires less

processing time in the online schedule, and it can be shifted in the time-interval of its released time and deadline. However, introducing extra speed will change rigid jobs into flexible jobs, which makes the whole problem much more complicated. Another further research direction is applying resource augmentation to feasible instances because obtaining a constant competitive ratio requires lots of extra resources. But it is possible that for feasible instances, to use fewer extra resources to obtain a constant competitive ratio.

References

[1] Nedeljko Vasić, Martin Barisits, Vincent Salzgeber, and Dejan Kostic. Making cluster applications energy-aware. In *Proceedings of the 1st workshop on Automated control for datacenters and clouds*, pages 37–42, 2009.

[2] Laura Cozzi, T Gould, S Bouckart, D Crow, TY Kim, C Mcglade, P Olejarnik, B Wanner, and D Wetzel. World energy outlook 2020. *vol*, 2050:1–461, 2020.

[3] Weiming Shi and Bo Hong. Resource allocation with a budget constraint for computing independent tasks in the cloud. In *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, pages 327–334. IEEE, 2010.

[4] Ana-Maria Oprescu and Thilo Kielmann. Bag-of-tasks scheduling under budget constraints. In *2010 IEEE second international conference on cloud computing technology and science*, pages 351–359. IEEE, 2010.

[5] Mihai Burcea, Wing-Kai Hon, Hsiang-Hsuan Liu, Prudence W. H. Wong, and David K. Y. Yau. Scheduling for electricity cost in smart grid. In Peter Widmayer, Yinfeng Xu, and Binhai Zhu, editors, *Combinatorial Optimization and Applications - 7th International Conference, COCOA 2013, Chengdu, China, December 12-14, 2013, Proceedings*, volume 8287 of *Lecture Notes in Computer Science*, pages 306–317. Springer, 2013.

[6] Fu-Hong Liu, Hsiang-Hsuan Liu, and Prudence W. H. Wong. Optimal nonpreemptive scheduling in a smart grid model. In Seok-Hee Hong, editor, *27th International Symposium on Algorithms and Computation, ISAAC 2016, December 12-14, 2016, Sydney, Australia*, volume 64 of *LIPIcs*, pages 53:1–53:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[7] Fu-Hong Liu, Hsiang-Hsuan Liu, and Prudence W. H. Wong. Greedy is optimal for online restricted assignment and smart grid scheduling for unit size jobs. In Evripidis Bampis and Nicole Megow, editors, *Approximation and Online Algorithms - 17th International Workshop, WAOA 2019, Munich, Germany, September 12-13, 2019, Revised Selected Papers*, volume 11926 of *Lecture Notes in Computer Science*, pages 217–231. Springer, 2019.

[8] Fu-Hong Liu, Hsiang-Hsuan Liu, and Prudence W. H. Wong. Non-preemptive scheduling in a smart grid model and its implications on machine minimization. *Algorithmica*, 82(12):3415–3457, 2020.

[9] Fu-Hong Liu, Hsiang-Hsuan Liu, and Prudence W. H. Wong. Greedy is optimal for online restricted assignment and smart grid scheduling for unit size jobs. *Theory Comput. Syst.*, 65(6):1009–1032, 2021.

[10] Mihai Burcea, Wing-Kai Hon, Hsiang-Hsuan Liu, Prudence W. H. Wong, and David K. Y. Yau. Scheduling for electricity cost in a smart grid. *J. Sched.*, 19(6):687–699, 2016.

[11] Mordechai Shalom, Ariella Voloshin, Prudence WH Wong, Fencol CC Yung, and Shmuel Zaks. Online optimization of busy time on parallel machines. *Theoretical Computer Science*, 560:190–206, 2014.

[12] ZA Lomnicki. A "branch-and-bound" algorithm for the exact solution of the three-machine scheduling problem. *Journal of the operational research society*, 16(1):89–100, 1965.

[13] Vincent Chau and Minming Li. Active and busy time scheduling problem: A survey. In *Complexity and Approximation*, pages 219–229. Springer, 2020.

[14] Edward G Coffman, Jr, Michael R Garey, David S Johnson, and Robert Endre Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4):808–826, 1980.

[15] Iren Suhami and Richard SH Mah. An implicit enumeration scheme for the flowshop problem with no intermediate storage. *Computers & Chemical Engineering*, 5(2):83–91, 1981.

[16] Susanne Albers and Matthias Hellwig. Online makespan minimization with parallel schedules. *Algorithmica*, 78(2):492–520, 2017.

[17] Lin Chen, Nicole Megow, and Kevin Schewior. An o(m)-competitive algorithm for online machine minimization. *SIAM Journal on Computing*, 47(6):2057–2077, 2018.

[18] Cynthia A Phillips, Cliff Stein, Eric Torng, and Joel Wein. Optimal time-critical scheduling via resource augmentation. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 140–149, 1997.

[19] Frederic Koehler and Samir Khuller. Busy time scheduling on a bounded number of machines. In *Workshop on Algorithms and Data Structures*, pages 521–532. Springer, 2017.

[20] Kirk Pruhs, Jiri Sgall, and Eric Torng. Online scheduling., 2004.

[21] Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *Journal of the ACM (JACM)*, 47(4):617–643, 2000.

[22] Nikhil Bansal, Ho-Leung Chan, Rohit Khandekar, Kirk Pruhs, Baruch Schieber, and Cliff Stein. Non-preemptive min-sum scheduling with resource augmentation. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 614–624. IEEE, 2007.