Multiprocessor Scheduling with Rejections: Theory and Practice

Student Name: Gaurav Gehlot
Roll Number: 2015030

BTP report submitted in partial fulfillment of the requirements
for the Degree of B.Tech. in Computer Science & Engineering
on 25^{th} April, 2019

BTP Track
Research track

BTP Advisor
Dr. Syamantak Das

Indraprastha Institute of Information Technology
New Delhi

Student's Declaration

I hereby declare that the work presented in the report entitled **Multiprocessor Scheduling with Rejections: Theory and Practice** submitted by me for the partial fulfillment of the requirements for the degree of *Bachelor of Technology* in *Computer Science & Engineering* at Indraprastha Institute of Information Technology, Delhi, is an authentic record of my work carried out under guidance of **Dr. Syamantak Das**. Due acknowledgements have been given in the report to all material used. This work has not been submitted anywhere else for the reward of any other degree.

..... **Place & Date: New Delhi, 25th April, 2019**
Gaurav Gehlot

Certificate

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

..... **Place & Date: New Delhi, 25th April, 2019**
Syamantak Das

Abstract

We consider the classical on-line load balancing problem of temporary tasks under restricted assignments. Each job can be processed only on a subset of machines, different jobs require different amount of processing time, and once a job is assigned to a machine it cannot be reassigned to another machine. The on-line load balancing problem is to assign each job to an appropriate machine such that the maximum load on any machine at any point of time is minimized.

Online algorithms are usually analyzed using the competitive ratio, which is defined as the ratio of the solution obtained by the online algorithm to that obtained by an offline adversary for the worst possible input sequence. The measure of analyzing online algorithms using the notion of competitive ratio is too bleak. Two popular approaches which have been used to deal with a variety of objective functions in case of scheduling problems have been that of "resource augmentation" [4] and "rejection model" [5].

In this project, our aim is to extend the work of Choudhury et al. [5] to the temporary jobs settings. As initial result, we analyze the standard greedy approach in rejection model without late rejections and arrive at the result that the competitive ratio for any on-line greedy algorithm is at-least $\Omega[(3m)^{2/3}/4]$ and the upper bound for the same problem is $[(3m)^{2/3}/2](1+o(1))$. Next, we consider the same problem and arrive at the result that any online algorithm would have competitive ratio at-least $\Omega(\sqrt{2m}/2)$. At last, we consider the same problem but this time with late rejections allowed and show that lower bound for this problem is at-least $\Omega(1/\varepsilon - 1)$ where ε is the fraction of jobs which can be rejected.

Keywords: Jobs, machines, restricted assignments, load balancing, rejection, greedy, algorithm, bound, competitive ratio.

Acknowledgments

I would like to thank my advisor, Dr. Syamantak Das, for providing me with the wonderful opportunity to work under him. He has been a constant support to me throughout the working of the project and has guided me on various issues during the project. Without his guidance and support, this work wouldn't have been possible.

Work Distribution

A brief description of work done during the duration of Monsoon semester '18.

Understanding scheduling problems: Aug '18 - Sep '18
Understanding the basics of scheduling problems and load balancing algorithm for permanent jobs with rejections.

Implementing load balancing algorithm and verifying the results: Sep '18 - Oct '18
Implementation of the load balancing algorithm for permanent jobs with rejections was done in C++ and the competitive ratio was verified for the worst possible sequence.

Greedy algorithm without late rejections: Oct'18 - Nov '18
The greedy algorithm of load balancing with temporary jobs was considered under the rejection model(without late rejections). The lower bound and upper bound for the same were calculated.

A brief description of work done during the duration of Winter semester '19.

Any Online algorithm without late rejection: Jan'19 - Feb '19
The tight bound was calculated for any online algorithm of load balancing with temporary jobs under the rejection model(without late rejections).

Any Online algorithm with late rejections: Mar '19 - Apr '19
The lower bound was calculated for any online algorithm of load balancing with temporary jobs under the rejection model(with late rejections).

Contents

1 Introduction **1**

1.1 Preliminaries 1

1.1.1 Job Classification and Restricted Assignments 1

1.1.2 Offline and Online Setting 1

1.1.3 ε-Rejection Model 2

1.2 Formal Description of the Problem 2

1.3 Notations 3

1.4 Results 3

1.4.1 Previous Results 3

1.4.2 Our Results 4

2 Load Balancing with Permanent Jobs **5**

2.1 Algorithm 5

2.2 Analysis and results 6

3 Load Balancing with Temporary Jobs and Immediate Rejection **7**

3.1 Upper bound for Greedy Algorithm 8

3.2 Lower bound for Greedy Algorithm 8

3.3 General Upper bound 10

3.4 General Lower Bound 10

3.5 Analysis on number of jobs for Greedy Algorithm 12

4 Load Balancing with Temporary Jobs and Late Rejection **14**

4.1 Lower bound 14

5 Future Work **16**

Bibliography **17**

Chapter 1

Introduction

In a local area network devices such as computers, I/O devices, workstations etc. are connected to each other through one or more gateways. Communication jobs , arrive and disappear at arbitrary times at a device requesting for a certain guaranteed bandwidth. The jobs must be assigned to one of the gateways connected to the device and service should begin immediately. The objective is to assign jobs to gateways in an appropriate way such that the maximum load at any gateway at any point of time is minimized. We consider a similar kind of problem, although with rejections. The on-line algorithm is allowed to reject up to ε-fraction of the total number of jobs(job arrivals).

1.1 Preliminaries

1.1.1 Job Classification and Restricted Assignments

There are two independent criteria which are used to classify jobs. The first is according to how long a job runs: The jobs which start at arbitrary times but continue forever are called permanent jobs, while the jobs that start at some time t and end after or on t are called temporary jobs. The second is according to set of machines on which a jobs can run: If a job can be assigned to any machine, it is termed as unrestricted assignments where as if it can only be assigned to a proper subset of the machines, it is termed as restricted assignments. We study the the case of restricted assignments with temporary jobs.

1.1.2 Offline and Online Setting

In the offline setting, the entire input, which includes the release time of the jobs and their sizes, is known to the scheduler right at the beginning. On the other hand, in an online setting, information about a job are revealed only at its release date. Such information might or might not include the total service time(duration of job) required. Further, the sequence in which the jobs are presented need not be arbitrary or random but can be adversarial as well.

1.1.3 ε-Rejection Model

The ε-Rejection Model allows the online algorithm to reject upto ε fraction of total number of jobs (job arrivals). On the other hand, the offline algorithm or OPT has to schedule all the jobs. There can be two type of rejection policy that can be considered under this type of model:

- Immediate-rejection: In immediate rejection the online algorithm needs to decide whether it wants to schedule the job or reject it on the arrival of the job itself. Once a job is scheduled, it can't be rejected.
- Late-rejection: Under late-rejection, even after scheduling the job, the online algorithm has the liberty to reject the job at some later point in time.

1.2 Formal Description of the Problem

Let M be a set of machines that is supposed to run a set of jobs. Each job j has a weight, or load, $w(j)$ associated to it, an arrival time $\tau(j)$, and a set $M(j) \subset M$ on which it can be run. Lets denote the fraction of jobs that an on-line assignment algorithm can reject by ε.
Upon arrival, each job can be rejected if after rejecting it, the fraction of rejected jobs will be less than ε, otherwise the job must be assigned to one of the machines which are capable of running it and the machine should start running it immediately, and should continue to run it until the job departs. The job once assigned, can't be reassigned. The option of rejecting jobs is only available for on-line algorithm whereas the offline algorithm must run all the jobs.
Let's denote the sum of weights(total load) on machine i at time t by $L_i(t)$. Let σ be a sequence of jobs arrival and departures and let $|\sigma|$ be the last time when a job arrives. Then the cost of an assignment algorithm A on the sequence σ is defined as:

$$C_A(\sigma) = max_{0 \leq t \leq |\sigma|; i \in M} \; L_i(t)$$

An optimal offline assignment algorithm, denoted by OPT, knows about the entire sequence σ and assigns jobs in such a way that minimizes it cost where as the online assignment algorithm, denoted by A, assign job j arriving at time $\tau(j)$ knowing only about $w(j), M(j)$, the present state of machines and past. *Note that the offline algorithm does not have the liberty of rejecting jobs but the on-line algorithm can reject jobs up to ε fraction.*
The competitive ratio of an online algorithm is defined as the maximum value of $C_A(\sigma)/C_{OPT}(\sigma)$ over all the possible sequences σ. The goal is to minimize this competitive ratio.

1.3 Notations

Following are the notations that we have used or plan to use.

- M is the set of all machines. $|M| = m$.
- ε is the fraction of jobs which the on-line algorithm can reject. $0 \leq \varepsilon < 1$
- Job j has a weight $w(j)$, an arrival time $\tau(j)$, and a set $M(j) \subset M$ on which it can be run.
- $T_i(t)$ is the set of jobs run (by the on-line algorithm) on machine i at time t.
- $T(t) = \underset{1 \leq i \leq m}{\cup} T_i(t)$ is the set of all jobs run by the on-line algorithm at time t.
- $L_i(t)$ is the load on machine i at time t. $L_i(t) = \sum_{j \in T_i(t)} w(j)$.
- $m(j) \in M(j)$ is the machine on which OPT (the optimum offline algorithm) runs job j. Thus for $S \subset M$, $m^{-1}(S)$ is the set of jobs assigned by OPT to machines in S.
- α is the competitive ratio that the on-line algorithm targets to achieve.

1.4 Results

1.4.1 Previous Results

For permanent jobs, restricted assignments, Azar et al. [3] showed that the competitive ratio of the greedy algorithm is precisely $\theta(log(m))$ and that no algorithm can do better. For the same problem but this time with ε-rejection model, Choudhury et al. [5] gave an algorithm with the competitive ratio $O(\log^2 1/\varepsilon)$ and also showed that for any(deterministic) online immediate dispatch algorithm the competitive ratio will be at-least $\Omega[log(1/\varepsilon)]$. For temporary jobs, restricted assignments, Y. Azar et al. [1] showed that the competitive ratio of the on-line greedy algorithm for this problem is $[(3m)^{2/3}/2](1 + o(1))$. Although the analysis of [5] doesn't generalize to temporary jobs but it is natural to ask that whether it is possible to obtain a greedy or randomized greedy algorithm for temporary jobs with restricted assignments under ε-rejection model which has the competitive ratio as function of ε. We answer this question negatively and show that the competitive ratio would be at-least $\Omega[(3m)^{2/3}/4]$ for greedy algorithm without late-rejection and at-least $\Omega(\sqrt{2m}/2)$ for any online algorithm without late-rejection. We also show that with late-rejection, the competitive ratio for any online algorithm would be at-least $\Omega(1/\varepsilon - 1)$.

Researcher	**Jobs type**	**Model**	**Result**
Azar et al. [3]	Permanent Jobs	No-rejection	$\theta(log(m))$
Choudhury et al. [5]	Permanent Jobs	ε-rejection	$\Omega(\log^2 1/\varepsilon)$
Y. Azar et al. [1]	Temporary Jobs	No-rejection	$[(3m)^{2/3}/2]\ (1+o(1))$
Our Result	Temporary Jobs	ε-rejection with immediate rejection	$\Omega(\sqrt{2m}/2)$
Our Result	Temporary Jobs	ε-rejection with late rejection	$\Omega(1/\varepsilon - 1)$

1.4.2 Our Results

(1) In the case of restricted assignments with temporary jobs under ε-rejection model without late-rejection, any greedy on-line assignment algorithm which can reject at most ε fraction of jobs would have competitive ratio at-least $\Omega[(3m)^{2/3}/4]$.

(2) In the case of restricted assignments with temporary jobs under ε-rejection model without late-rejection, any on-line assignment algorithm which can reject at most ε fraction of jobs would have competitive ratio at-least $\Omega(\sqrt{2m}/2)$.

(3) In the case of restricted assignments with temporary jobs under ε-rejection model with late-rejection, any on-line assignment algorithm which can reject at most ε fraction of jobs would have competitive ratio at-least $\Omega(1/\varepsilon - 1)$.

Chapter 2

Load Balancing with Permanent Jobs

Choudhury et al. [5] gave the algorithm for load balancing with permanent jobs in restricted assignments under ϵ-rejection model which was $\Omega(\log^2 1/\epsilon)$ competitive. We wanted to check how the algorithm performs in real life scenario i.e. we wish to compare the competitive ratio of real life test cases to the worst case competitive ratio. The competitive ratio of $\Omega(\log^2 1/\epsilon)$ by Choudhury et al. [5] was a significant achievement because the result was independent of m unlike the results obtained by Azar et al. [3] before this in the setting *without* ϵ-rejection model.

2.1 Algorithm

The general algorithm which is given by Choudhury et al. [5] has 2 stages which are oblivious to each other. The details of stage 1, stage 2 are given in Figure 2.1 (a) and (b) respectively and the final algorithm is given in Figure 2.1 (c). The algorithm becomes very simple in case of unit size jobs and the details of it are given in Figure 2.1 (d).

Stage 1: When job j of class l arrives at time t:

Let $i^\star = \underset{i \in S_j}{\operatorname{argmin}}\, \texttt{load}_{i,l}(t)$

If $\texttt{load}_{i^\star,l}(t) + p_j \leq \alpha T^\star$ **Then**

Assign j to $i^\star$ **Else**

Reject j

(a) Stage 1

Stage 2: Pruning machine i at time t

Let $\texttt{load}_i(t) = \sum_l \texttt{load}_{i,l}(t)$

While $\texttt{load}_i(t) > 2\alpha T^\star$

Reject the largest job assigned to i

(Break ties in an arbitrarily chosen but fixed way)

(b) Stage 2

Load Balance:

1. Initialize $T^\star \leftarrow p_j$, where j is the first released job.
2. **Repeat**

(i) Run $\mathcal{A}(T^\star)$ on the jobs which arrive next.

(ii) If the algorithm rejects more than ε-fraction of the jobs which arrived after the time it started running,

Stop the algorithm and update $T^\star \leftarrow 2T^\star$.

(c) Load balancing Algorithm

When job j arrives at time t:

Let $i^\star = \underset{i \in S_j}{\operatorname{argmin}}\, \texttt{load}_i(t)$

If $\texttt{load}_{i^\star}(t) + 1 \leq \alpha T^\star$ **Then**

Assign j to $i^\star$ **Else**

Reject j

(d) Algorithm for unit size jobs

Figure 2.1: Algorithms for load balancing with permanent jobs

2.2 Analysis and results

To analyze the algorithm, it was implemented in C++ language and the test was carried out on the worst case with unit size jobs. The results of the test are documented in the table below in Figure 2.2.

Machines(m)=65536				
Alpha = (log(1/e))	**epsilon(e)**	**Max Load**	**Total Jobs**	**Jobs Rejected**
1	1	1	65535	32767
2	0.25	2	65535	16383
3	0.125	3	65535	8191
4	0.0625	4	65535	4095
5	0.03125	5	65535	2047
6	0.015625	6	65535	1023
7	0.0078125	7	65535	511
8	0.00390625	8	65535	255
9	0.001953125	9	65535	127
10	0.0009765625	10	65535	63

Figure 2.2: Output of implementation of algorithm

A graph(see Figure 2.3) was also made between ϵ and the maximum load on the on-line assignment algorithm. Upon analysis of which it can be seen that the graph is following the logarithmic scale verifying the result of $\alpha = \log_2 1/\epsilon$.

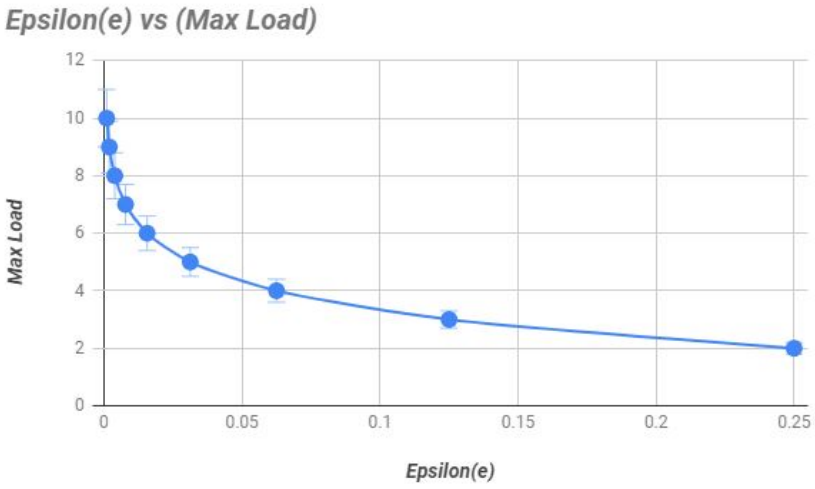

Figure 2.3: Graph between ϵ and Maximum Load

Chapter 3

Load Balancing with Temporary Jobs and Immediate Rejection

In this section, we consider the case of temporary jobs in restricted assignment setting under ϵ-rejection model without late-rejection. Choudhury et al. [5] paper uses a greedy approach with permanent jobs under ϵ-rejection model but we show that it is not good in the case of temporary jobs even with rejections. We derive that the lower bound of any greedy on-line assignment algorithm is at-least $\Omega[(3m)^{2/3}/4]$ and state the upper bound as $[(3m)^{2/3}/2](1 + o(1))$. We also do analysis regarding the number of jobs required to make the on-line algorithm reach a fixed competitive ratio$(\alpha/2)$and express it as a function of α.

We start with the assumption that the on-line algorithm knows the value of offline optimum solution, denoted by T^*. Removing this assumption requires standard ideas in online algorithms which we do not need to worry about. Here, α is the competitive ratio, which the on-line algorithm targets to achieve.

Algorithm(A) is a greedy algorithm which schedules the arriving job on the machine with minimum load and breaks the ties arbitrarily. A is defined as follows:

When job j arrives at time t:
    $i^* = \underset{i \in M}{\text{argmin}}\ L_i(t)$
    If $L_{i^*}(t) + w(j) \leq \alpha T^*$:
        Schedule j on i^*
    Else:
        Reject j

Clearly, the load on any machine$(L_i(t))$ is always less than αT^* because upon arrival of any job j, we schedule it on machine with minimum load(i^*) only when $L_{i^*}(t) + w(j) \leq \alpha T^*$, otherwise we reject it.

3.1 Upper bound for Greedy Algorithm

Theorem 3.1. *For any sequence σ of job arrivals and departures, the greedy or randomized greedy on-line assignment algorithm which can reject at most ϵ fraction of jobs is $[(3m)^{2/3}/2](1+o(1))$ competitive.*

Proof. This theorem follows from the result obtained by Y. Azar et al. [1] which stated that: For any sequence σ of job arrivals and departures, the greedy on-line assignment (without rejections) is $[(3m)^{2/3}/2](1+o(1))]$ competitive". [1]

3.2 Lower bound for Greedy Algorithm

Lemma 3.2.1. *There exists a finite sequence σ of job arrivals and departures which can make any greedy or randomized greedy on-line assignment algorithm(A) which targets competitive ratio$(\alpha) < [(3m)^{2/3}/4]$ to have load sequence: $\alpha T^*, (\alpha T^* - 1), (\alpha T^* - (1+2)), ...$
$((\alpha T^* - (1+2.. + (q-1))$ where $\binom{q}{2} < \alpha T^*$.*

Proof. We build a sequence σ with following properties:

(1) There are m jobs running at time $|\sigma|$. $T(|\sigma|) = m$. [1]

(2) The maximum offline load at all times t$(0 \leq t \leq |\sigma|)$, is 2. This also means that $T^* = 2$.

(3) If at time $|\sigma|$ A uses machine i at all$(L_i(t) > 0)$ then A runs on i, the job that OPT is running on i. This will be for a simple reason that the job can't be run anywhere else i.e. $M(m^{-1}(i)) = \{i\}$. [1]

Let $S(t) = (S_1(t), S_2(t), ..., S_m(t))$ be the sorted non-increasing sequence of loads that A has on machines at time t, and let $s_i(t)$ be the machine corresponding to load $S_i(t)$. [1] Lets also define a property(P) on S as:

(P) For all k, such that $S_k(|\sigma|) > 0$, $S_{k-1}(|\sigma|) - S_k(|\sigma|) \geq k - 1$.

We build the sequence σ, in the following way:

(1) **Initial Step:** For each machine $i \in M$ a job, a new job j_i arrives with $M(j_i) = \{i\}$. Note that after this step, σ satisfies the three properties listed above.

(2) If $S_1(t) \geq \alpha T^*$ we stop, else we build a sequence σ' in the following steps to extend σ.

(3) Let $k \geq 2$ be the minimum value for which $S_k(|\sigma|) > 0$ and $S_{k-1}(|\sigma|) - S_k(|\sigma|) = d \leq k-2$. (It is guaranteed that such a value always exist if $S_1(t) < \alpha T^*$, we show the proof for this later).

(4) For $1 \leq i \leq d$, job $m^{-1}(s_i)$ departs. The result: load decreases on s_1, s_2,....,s_d by 1. [1]

(5) For $1 \leq i \leq d$, a new job j_i arrives with $M(j_i) = \{s_i, s_k\}$. A assigns all of them to s_k due to its greedy nature(s_k is always less loaded than alternative machine) where as OPT assigns j_i to s_i. After this step, L_k becomes equal to L_{k-1}. [1]

(6) Jobs $m^{-1}(s_k)$ and $m^{-1}(s_{k-1})$ departs. [1]

(7) Three additional jobs j_a, j_b, j_c arrives with $M(j_a) = \{s_k\}$, $M(j_b) = \{s_k, s_{k-1}\}$ and $M(j_c) = \{s_{k-1}\}$. A assigns j_a to s_k and j_b, j_c to s_{k-1}. OPT assigns j_a and j_b to and s_k and j_c to s_{k-1}. (At this point and only at this point OPT has load 2 on s_k, otherwise the load is 1 on all the machines). [1]

(8) Jobs j_i, $1 \leq i \leq d$ departs and then arrive again renamed $j_i^{'}$ with $M(j_i^{'}) = \{s_i\}$. Job j_a departs. *At this point OPT again has load 1 on all the machines.* [1]

(9) Each task h run by A on server s_k departs and then arrives again, renamed h' with $M(h') = m(h)$. [1]

(10) We set $\sigma = \sigma + \sigma^{'}$ and repeat steps 2 to 9 again until $S_1(t) = \alpha T^*$

Note that after every extension of σ the three properties 1-3 holds and each extension also increases the lexicographical value of $S(|\sigma|)$, and as long as property (P) doesn't hold, it can be further extended. Therefore after some finite number of extensions we will reach the state where $S_1(t) = \alpha T^*$.
After building the sequence σ, OPT has load 1 on all the machines where as A has load sequence$(S|\sigma|)$ as:

$$\alpha T^*, (\alpha T^* - 1), (\alpha T^* - (1+2)), \ldots\ldots, ((\alpha T^* - (1+2+\ldots+(q-1))$$

where q is the number of active machines having $load > 0$ at time $|\sigma|$ and $\binom{q}{2} < \alpha T^*$.
We know that there are m jobs at $|\sigma|$, therefore:

$$\alpha T^* + (\alpha T^* - 1) + (\alpha T^* - (1+2)) + \ldots\ldots + ((\alpha T^* - (1+2+\ldots+(q-1)) \geq m,$$

Note that, $\binom{q}{2} < \alpha T^*$. Therefore, we get $\alpha T^* \geq [(3m)^{2/3}/2](1 + o(1)$ but T^* is equal to 2 and $\alpha < \Omega(3m)^{2/3}/4$ which means $\alpha T^* < \Omega[(3m)^{2/3}/2]$ i.e. it is always possible to extend the sequence σ until $S_1(t) = \alpha T^*$.

Theorem 3.2. *Any greedy online assignment algorithm which can reject at most ϵ fraction of jobs will have competitive ratio at-least $\Omega[(3m)^{2/3}/4]$.*

Proof. If $\alpha \geq [(3m)^{2/3}/4]$ then the theorem is trivially satisfied. We need to consider the case where $\alpha < (3m)^{2/3}/4$. In this case, we show that there exists a sequence of job arrivals and departures which will cause A to reject unbounded number of jobs. For that, from Lemma 3.2.1, we know that there exist a sequence(σ) of job arrival and departures which will make the load sequence on A at $|\sigma|$ as $\alpha T^*, (\alpha T^* - 1), (\alpha T^* - (1+2)), ...((\alpha T^* - (1+2..+(q-1))$ whereas the OPT will have load 1 on all the machines.

After building σ according to the procedure given in *Lemma* 3.2.1, we extend it by the following steps:

(1) Release a job j_r with $M(j_r) = \{s_1\}$. OPT will schedule it on s_1 whereas A will reject it because $L_{s_1}|(\sigma)| = \alpha T^*$. The loads in A do not change whereas in OPT, on s_1 load becomes 2 and rest of the machines have load 1 only.

(2) Depart j_r. The load on all the machines of OPT becomes 1 where as the loads in A do not change.

(3) Repeat steps (1), (2) and (3).

With every execution of this procedure, the fraction of rejected jobs increases. Therefore it is always possible to make: *Fraction of rejected jobs* $> \varepsilon$. Hence, the competitive ratio must be at-least $\Omega[(3m)^{2/3}/4]$.

3.3 General Upper bound

Theorem 3.3. *For any sequence σ of job arrivals and departures, on-line assignment algorithm which can reject at most ε fraction of jobs is $[2\sqrt{m}+1]$ competitive.*

Proof. This theorem follows from the analysis of Robin-Hood algorithm [2]. The algorithm has a competitive ratio which is at-most $[2\sqrt{m}+1]$. [2]

3.4 General Lower Bound

Lemma 3.4.1. *There exists a finite sequence σ of job arrivals and departures which can make any on-line assignment algorithm(A) which targets competitive ratio$(\alpha) < (\sqrt{2m}/2)$ to have load sequence: $\alpha T^*, (\alpha T^* - 1), (\alpha T^* - 2), ...(\alpha T^* - (q-1))$ where $q \leq \alpha T^*$.*

Proof. We build a sequence σ with the same three properties as we used in *Lemma* 3.2.1. Also, let $S(t)$, $s_i(t)$, denote the same things as in *Lemma* 3.2.1.
Lets, define a property $(P2)$ on S as:

(P2) For all k, such that $S_k(|\sigma|) > 0$, $S_{k-1}(|\sigma|) - S_k(|\sigma|) \geq 1$.

We build the sequence σ, in the following way:

(1) **Initial Step:** For each machine $i \in M$ a job, a new job j_i arrives with $M(j_i) = \{i\}$. Note that after this step, σ satisfies the three properties listed above.

(2) If $S_1(t) \geq \alpha T^*$ we stop, else we build a sequence $\sigma^{'}$ in the following steps to extend σ.

(3) Let $k \geq 2$ be the minimum value for which $S_k(|\sigma|) > 0$ and $S_{k-1}(|\sigma|) - S_k(|\sigma|) = d \leq 1$. (It is guaranteed that such a value always exist if $S_1(t) < \alpha T^*$, we show the proof for this later).

(4) Jobs $m^{-1}(s_k)$ and $m^{-1}(s_{k-1})$ departs. This results in the decrease in the load of both the machines by 1. [1]

(5) A new job j_1 arrives with $M(j_1) = \{s_k, s_{k-1}\}$. [1]
(6) This step depends upon whether A assigns j_1 to s_k or s_{k-1}. [1]

- A assigns j_1 to s_k: OPT assigns j_1 to s_{k-1} and a new job j_2 arrives with $M(j_2) = \{s_k\}$. All jobs J which are scheduled on s_{k-1}(in A) depart and arrive again, with $M(j) = m(j)(j \in J)$. Load on s_k increases by 2 from before(before step 4) and hence the lexicographical value of S increases from before.

- A assigns j_1 to s_{k-1}: OPT assigns j_1 to s_k and a new job j_2 arrives with $M(j_2) = \{s_{k-1}\}$. All jobs J which are scheduled on s_k(in A) depart and arrive again, with $M(j) = m(j)(j \in J)$. Load on s_{k-1} increases by 2 from before(before step 4) and hence the lexicographical value of S increases from before.

(7) We set $\sigma = \sigma + \sigma^{'}$ and repeat steps 2 to 7 again until $S_1(t) = \alpha T^*$

Note that after every extension of σ the three properties 1-3 holds and each extension also increases the lexicographical value of $S(|\sigma|)$, and as long as property (P2) doesn't hold, it can be further extended. Therefore after some finite number of extensions we will reach the state where $S_1(t) = \alpha T^*$.
After building the sequence σ, OPT has load 1 on all the machines whereas A has load sequence($S|\sigma|$) as:

$$\alpha T^*, (\alpha T^* - 1), (\alpha T^* - 2), \ldots\ldots, (\alpha T^* - (q-1))$$

where q is the number of active machines having $load > 0$ at time $|\sigma|$ and $q \leq \alpha T^*$.
We know that there are m jobs at $|\sigma|$, therefore:

$$\alpha T^* + (\alpha T^* - 1) + (\alpha T^* - 2) + \ldots\ldots + (\alpha T^* - (q-1)) \geq m,$$

By using $q \leq \alpha T^*$, we get $\alpha T^* \geq \sqrt{2m}$ but $T^* = 2$ and $\alpha < (\sqrt{2m}/2)$ which means $\alpha T^* < \sqrt{2m}$ i.e. it is always possible to extend the sequence σ until $S_1(t) = \alpha T^*$.

Theorem 3.4. *Any deterministic online assignment algorithm which can reject at most ε fraction of jobs will have competitive ratio at-least $\Omega(\sqrt{2m}/2)$.*

Proof. If $\alpha \geq (\sqrt{2m}/2)$ then the theorem is trivially satisfied. We need to consider the case where $\alpha < (\sqrt{2m}/2)$. In this case, we show that there exists a sequence of job arrivals and departures which will cause A to reject unbounded number of jobs. For that, from Lemma 3.4.1, we know that there exist a sequence(σ) of job arrival and departures which will make the load sequence on A at $|\sigma|$ as $\alpha T^*, (\alpha T^* - 1), (\alpha T^* - 2), ...(\alpha T^* - (q-1))$ whereas the OPT will have load 1 on all the machines.

After building σ according to the procedure given in *Lemma* 3.4.1, we extend it by the following steps:

(1) Release a job j_r with $M(j_r) = \{s_1\}$. OPT will schedule it on s_1 whereas A will reject it because $L_{s_1}|(\sigma)| = \alpha T^*$. The loads in A do not change whereas in OPT, on s_1 load becomes 2 and rest of the machines have load 1 only.

(2) Depart j_r. The load on all the machines of OPT becomes 1 where as the loads in A do not change.

(3) Repeat steps (1), (2) and (3).

With every execution of this procedure, the fraction of rejected jobs increases. Therefore it is always possible to make: *Fraction of rejected jobs* $> \varepsilon$. Hence, the competitive ratio must be at-least $\Omega(\sqrt{2m}/2)$.

3.5 Analysis on number of jobs for Greedy Algorithm

In this section, we analyze the number of jobs required to make the on-line assignment algorithm reach the load sequence $\alpha T^*, (\alpha T^* - 1), (\alpha T^* - (1+2)), ...((\alpha T^* - (1+2.. + (q-1))$.

Let define an *operation* for $k \geq 2$ as: Taking two consecutive machines at time t such that $S_k(t) > 0$ and $S_{k-1}(t) - S_k(t) = d \leq k-2$ and then making $S_{k-1}(t+1) = S_{k-1}(t) + 1$ by using the steps 4-8 given in the build-up of sequence σ.

Then, number of job arrivals require to perform an *operation* with $S_{k-1}(t) - S_k(t) = d$ is equal to $2d + 3 + w^{'}(k) - 1$ where $w^{'}(k)$ denotes the value of $S_k(t)$ at time t just before the operation.

Let's look at a step by step application of build-up procedure stated in *Lemma* 3.2.1 from some arbitrarily initial state. The machines in red colors are the ones where property P is violated:

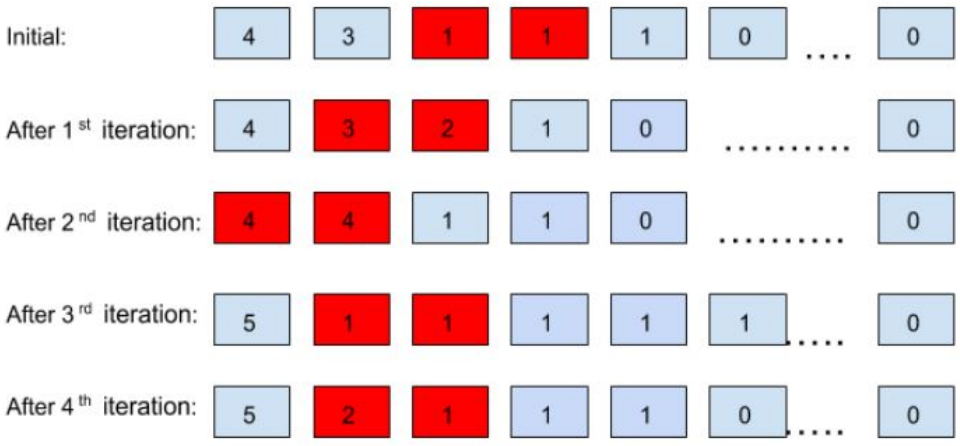

Figure 3.1: Pictorial representation of build-up of sequence σ

Lets denote the value of d that would have been at $k = i$ as D_i. Then for $\binom{q}{2} < \alpha T^*$ and $2 \leq i \leq q$:

$$D_i \leq (i-2)$$

Lets N_k^l denote the number of job arrivals required to make $S_k = l$ from the state where $S_i \leq 1$ for all $(i \geq k)$. Clearly the answer we are seeking for is $N_1^{\alpha T^*}$.
Note that we are calculating the upper bound on number of job arrivals, so we consider the worst case that when we perform an operation to increase the load on s_{k-1} the load on all the machines $s_i, (k \leq i \leq m)$ becomes less than or equal to 1. Therefore,

$$N_1^{\alpha T^*} = \sum_{1 \leq i_1 \leq \alpha T^* - 1} [2(0) + 3 + (i_1 - 1) + N_2^{i_1 - 0}]$$

$$N_2^{i_1} = \sum_{1 \leq i_2 \leq i_1 - 1} [2(1) + 3 + (i_2 - 2) + N_3^{i_2 - 1}]$$

$$N_3^{i_2 - 1} = \sum_{1 \leq i_3 \leq i_2 - 2} [2(2) + 3 + (i_3 - 3) + N_4^{i_3 - 2}]$$

$$\vdots$$

$$N_{q+1}^{i_q - q + 1} = \sum_{1 \leq i \leq i_q - q} [2(q) + 3 + o(1)]$$

The number of jobs required can be expressed as summation series with parameter α. Note that, $\binom{q}{2} < \alpha T^*$ therefore the number of jobs required are finite and depends purely on α.

Chapter 4

Load Balancing with Temporary Jobs and Late Rejection

In this section, we consider the problem of load balancing of temporary jobs in restricted assignment setting under ε-rejection model with late-rejection. We derive that the lower bound for any on-line assignment algorithm is at-least $\Omega(1/\varepsilon - 1)$ where ε is the maximum fraction of jobs which can be rejected. The lower bound proof is inspired from the proof of Theorem "The competitive ratio of any deterministic on-line algorithm is at-least $\Omega(min(m^{1/2}, T^{1/3}))$ in [6].

4.1 Lower bound

Theorem 4.1. *The competitive ratio of any deterministic on-line algorithm in restricted assignment setting with ε-rejection model and late-rejection is at-least $\Omega(1/\varepsilon - 1)$.*

Proof. Let $r = m^{1/2}/2$. We show that any deterministic on-line algorithm A must have a competitive ratio at-least $\Omega(1/\varepsilon - 1)$ in ε-rejection model with late-rejection. Consider an adversary who uses the following rules to build a sequence of r^3 jobs, each having a load of 1.

(1) Assume a discrete time series, exactly one job arrive at each time step. [6]

(2) The job sequence is presented in r^2 phases, where each phase has r jobs. [6]

(3) In phase $i(1 \leq i \leq r)$, the j^{th} job $(1 \leq j \leq r)$ can be scheduled on either machine j or machine $r + i$. [6]

Lets assume that online algorithm does its best and achieves symmetry with its load then the load on all the machines in A would be equal. Let denote that load by x, then by equating total load scheduled to minimum load which need to be scheduled we get:

$$(r + r^2)x = r^3(1 - \varepsilon)$$
$$\text{By solving we get, } x = r^2(1 - \varepsilon)/(1 + r)$$

Now, OPT can decide to schedule the jobs which are schedule on machine 1 to r in i^{th} phase on its machine $r + i$ and depart all the other jobs. So the maximum load on OPT(T^*) becomes $(r - x)$ as in every phase x amount of jobs are scheduled on $r + i$. Therefore competitive ratio α:

$$\alpha = x/(r - x)$$
$$\text{By solving it}(assuming <<< r) \text{ we get, } \alpha = 1/\varepsilon - 1$$

Note that, if online algorithm decides to schedule more than x jobs on any machine i ($1 \leq i \leq r$), then OPT can choose to limit its load to x and not schedule the extra jobs. In case of online algorithm deciding to schedule more than x jobs in any machine $r + i$, the competitive ratio gets trivially increased. Hence, in any case the online algorithm has competitive ratio at-least $\Omega(1/\varepsilon - 1)$.

Chapter 5

Future Work

(1) Finding an upper bound or an online algorithm for load balancing with temporary jobs in ε-rejection model with late-rejections whose performance matches with our lower bound, or improving the lower bound for the same problem.
(2) Considering the problem of temporary jobs in ε-rejection model with known duration i.e. where departure time is know as soon as job arrives.

Bibliography

[1] Azar, Y., Broder, A. Z., and Karlin, A. R. On-line load balancing. *Theor. Comput. Sci. 130*, 1 (1994), 73–84.

[2] Azar, Y., Kalyanasundaram, B., Plotkin, S. A., Pruhs, K., and Waarts, O. On-line load balancing of temporary tasks. In *Algorithms and Data Structures, Third Workshop, WADS '93, Montréal, Canada, August 11-13, 1993, Proceedings* (1993), pp. 119–130.

[3] Azar, Y., Naor, J., and Rom, R. The competitiveness of on-line assignments. In *Proceedings of the Third Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 27-29 January 1992, Orlando, Florida, USA.* (1992), G. N. Frederickson, Ed., ACM/SIAM, pp. 203–210.

[4] Bansal, N., Chan, H., Khandekar, R., Pruhs, K., Stein, C., and Schieber, B. Non-preemptive min-sum scheduling with resource augmentation. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings* (2007), IEEE Computer Society, pp. 614–624.

[5] Choudhury, A. R., Das, S., Garg, N., and Kumar, A. Rejecting jobs to minimize load and maximum flow-time. *J. Comput. Syst. Sci. 91* (2018), 42–68.

[6] Ma, Y., and Plotkin, S. A. An improved lower bound for load balancing of tasks with unknown duration. *Inf. Process. Lett. 62*, 6 (1997), 301–303.