Scheduling for Multi-Phase Parallelizable Jobs

Rahul Vaze
School of Technology and Computer Science
Tata Institute of Fundamental Research
Mumbai, India.
rahul.vaze@gmail.com

***Abstract*—With multiple identical unit speed servers, the online problem of scheduling jobs that migrate between two phases, limitedly parallelizable or completely sequential, and choosing their respective speeds to minimize the total flow time is considered. In the limited parallelizable regime, allocating k servers to a job, the speed extracted is $k^{1/\alpha}, \alpha > 1$, a sub-linear, concave speedup function, while in the sequential phase, a job can be processed by at most one server with a maximum speed of unity. A LCFS based algorithm is proposed for scheduling jobs which always assigns equal speed to the jobs that are in the same phase (limitedly parallelizable/sequential), and is shown to have a constant (dependent only on $\alpha > 1$) competitive ratio. For the special case when all jobs are available beforehand, improved competitive ratio is obtained.**

I. Introduction

In the presence of multiple servers, how to schedule *parallelizable* jobs to minimize the sum of their response times (called the flow time) is an incredibly important and analytically challenging problem, e.g. in large data centers. With multiple servers, the parallelizability of job is captured by the total speed assigned to it when processed by multiple servers simultaneously. Let the total number of servers be N, where each server can operate at the maximum speed of unity. Then, typically [1], [2], [3], [4], [5], [6], [7], [8], [9], if $k \leq N$ is the number of servers assigned to a job, the resulting speed obtained is $s(k) = k^{1/\alpha}$. Depending on α (called the speed-up exponent), i) if $\alpha = 1$, the job is called fully parallelizable, otherwise if $\alpha > 1$, its called limitedly parallelizable, while if $\alpha = \infty$ for $\forall\ k > 1$ and $\alpha = 1$ for $k \leq 1$, it is called sequential.

In most practical settings [10], [11], [12], [13], [14] each job does not necessarily have a single phase of parallelizability, but migrates between different phases at different times during its execution. For example in a MapReduce framework [14], initially, jobs have full/limited parallelizability, while in the concluding stages they become sequential. Given practical considerations as described in detail in [15], it is reasonable to consider the case of jobs having either limited parallelizability [2] (called elastic phase), or are sequential (called in-elastic phase), where $2 \leq \alpha \leq 3$ is the most relevant regime for limited parallelizability.

Thus, in this paper, we consider the online problem of scheduling jobs and how many servers to allocate to each job being processed to minimize the flow time, where each job has two possible types of phases of parallelizability, either elastic or in-elastic, and where jobs arrive at arbitrary times, have arbitrary number of elastic and in-elastic phases, and have arbitrary job sizes for each phase. To quantify the performance of an online algorithm, we consider the metric of competitive ratio, that is defined as the ratio of the flow time of the online algorithm and the optimal offline algorithm OPT (that knows the entire input sequence in advance) maximized over all possible inputs (worst case).

A. Prior Work

1) Single Phase: With limited parallelizability, the single phase scheduling problem of finding how many servers to allocate to each job that minimizes the flow time is challenging, and has been an object of immense interest [1], [2], [3], [4], [5], [6], [9]. With limited parallelizability, the single phase scheduling problem has been considered for two models i) the combinatorial discrete allocation model [3], where an integer number of servers are assigned to any job, and ii) the continuous allocation model [4], [5], [6], [1], [2], [9], that treats the N servers as a single resource block which can be partitioned into any size and assigned to any job. In the continuous allocation model, for the online case where jobs arrive over time, [9] proposed a constant competitive algorithm that only depends on the exponent α, while an optimal algorithm has been derived in [2] when all jobs are available at time 0. In practice, some of the methods for server allocation include packing based [7], and resource reservation algorithms [16]. Heuristic policies with only numerical performance analysis can be found in [8].

2) Multiple Phases: The multiple phase scheduling problem has primarily been considered in the continuous allocation model [4], [5], [6], where there are arbitrary number of phases with arbitrary speed-up exponents α for each phase. In this line of work, mostly the non-clairvoyant setting (the algorithm is not aware of the remaining size of the jobs or the exponent α of the current/future phases), with few exceptions where clairvoyant setting has been studied [17], [18]. The competitive ratio of any non-clairvoyant online algorithm (both deterministic and randomized) is known to be at least $\sqrt{n}$ (n is the total number of jobs) [4], when there are arbitrary number of phases with different exponents α.

In light of the lower bound, resource augmentation is considered, where an algorithm is allowed more resources than the optimal offline algorithm. Algorithms with constant competitive ratios have been derived as a function of the resource augmentation factor [4], [5]. In particular, algorithm

EQUI that assigns equal speed to all jobs (without knowing even the current phase index for each job) has a constant competitive ratio when given double the number of servers compared to the OPT [4]. A more refined competitive ratio result with resource augmentation was derived in [5]. Surprisingly, for the special case, where all phases are strictly ρ sub-linear for any $\rho > 0$, where the speed function $s(k)$ (speed assigned to job when allocated k servers) satisfies the relation $\frac{s(k_2)}{s(k_1)} \leq \left(\frac{k_2}{k_1}\right)^{1-\rho}$ whenever $k_1 \leq k_2$, EQUI has a competitive ratio of $2^{1/\rho}$ against a clairvoyant optimal offline algorithm without any resource augmentation [4]. Notably, the in-elastic phase considered in this paper is not strictly ρ sub-linear.

From a practical point of view, the two phase problem is more relevant, and for which heuristic policies, e.g., the phase-aware FCFS [19] that schedules jobs in their arrival order, while assigning at most speed 1 to a job that is in its in-elastic phase, have been proposed. Some partial results have been derived in [20] for the two-phase scheduling problem. In very recent work, [15] characterized an optimal scheduling policy, for the two-phase scheduling problem as studied in this paper, however, with two strong assumptions, i) the size of jobs in the elastic and in-elastic phases are exponentially distributed with the same parameters for all jobs, and are independent of each other, and ii) the job always completes when it is in its in-elastic phase. We avoid all these assumptions in this paper, by letting the job sizes in each phase to be arbitrary, and the first and the last phase of a job can either be elastic or in-elastic.

B. Our contributions

For the two-phase scheduling problem, we propose an algorithm called FRACTIONAL-LCFS that processes a fraction of the outstanding jobs that have arrived most recently, and a subset of inelastic jobs, where each type of scheduled job is executed with equal speed. The exact choice is more refined and detailed in Section IV. The algorithm is semi non-clairvoyant that disregards the remaining job sizes of all remaining phases (even though they are known), and only uses the information about the current phase each job is in.

The choice of which jobs to process by the algorithm is defined by the number of jobs in each of the two-phases. Compared to the algorithm [15] that always prioritises jobs that are in their in-elastic phases, our algorithm prioritises jobs that are in their in-elastic phase only when there are sufficiently many of them and the total number of jobs is less compared to the total number of servers.

We show that FRACTIONAL-LCFS has a constant competitive ratio (derived in Theorem 1) that depends only on the speed-up exponent $\alpha > 1$ and not on system parameters such as the total number of jobs, and their respective sizes, and the number of servers. This result overcomes fundamental challenge left open in the literature for the considered problem, where speed augmentation was needed to prove constant competitiveness [4]. It is worth mentioning that we do not get any meaningful competitive ratio when $\alpha = 1$ (fully-parallizable jobs), since for this case, a lower bound of $n^{1/3}$ (n is the total number of jobs) on the competitive ratio is known [21] for any deterministic algorithm that is unaware of the remaining sizes of the jobs, similar to the algorithm FRACTIONAL-LCFS.

We also consider the simpler setting where all jobs are available at time 0. Similar to the online jobs arrival case, in this case also, we propose an algorithm that makes three different choices on which jobs to schedule depending on the number of jobs in the system and the number of servers. Moreover, it assigns equal speed to all jobs that are being processed that belong to the same phase. Compared to the online jobs arrival case, we get a significantly improved competitive ratio bound in this simpler case provided in Theorem 3. It is worth recalling that an optimal algorithm for the single phase problem where all jobs are available at time 0 has been derived in [2], however, no such result is known for the two-phase problem.

In addition to the analytical results, we also present average-case simulation results to illustrate the actual performance of the proposed algorithm. We compare the performance of our proposed algorithm with EQUI, the inelastic first algorithm [15], as well as the phase aware FCFS [19], and observe that the performance of our algorithm is comparable or better than EQUI and the inelastic first algorithm, while outperforming phase aware FCFS always.

II. SYSTEM MODEL

Let there be N parallel and identical servers, each with speed 1. The set of jobs is denoted by $\mathcal{J}$, where a job $j \in \mathcal{J}$ arrives at time a_j. Similar to [4], [2], we consider the continuous allocation model, where N is treated as a single resource block which can be divided into chunks of arbitrary sizes and allocated to different jobs.

Each job j at any time can be in one of two phases, called elastic or in-elastic. The sizes of job j in the a^{th} elastic and b^{th} in-elastic phase are w_{je}^a and $w_{j\iota}^b$, respectively. Moreover, let A_j and B_j be the total number of elastic and in-elastic (interleaved) phases required for each job, respectively. The first and the last phase of any job can be either of the two phases. We consider the online setting, where an algorithm has only causal information about jobs, i.e. any job's phases and their respective sizes are revealed only once it arrives.

In the elastic phase, any job is parallelizable with concave speedup, i.e., if job j is allotted $k_j(t)$ number of servers at time t, then the service rate experienced by job j at time t is $s_j(t) = P(k_j(t)) = k_j(t)^{1/\alpha}$, where $\alpha > 1$. Since we are considering the continuous allocation model, $k_j(t) < 1$ is possible. Similar to [2], [9], [15], we let $s(k) = k_j(t)^{1/\alpha}$ even when $k_j(t) < 1$.

In the in-elastic phase, each job can be processed by at most one server, and equivalently can be processed at speed of at most 1. Moreover, for any job j, it transitions from the elastic to in-elastic phase or vice versa only when its total work w_{je}^a or $w_{j\iota}^b$ in the current phase is complete.

A job j is defined to be complete at time d_j, if d_j is the earliest time at which total $\sum_{a\leq A_j} w^a_{je} + \sum_{b\leq B_j} w^b_{j\iota}$ amount of work has been completed for job j, and the objective is to minimize the flow time

$$\min F = \sum_{j\in\mathcal{J}}(d_j - a_j) = \int n(t)dt \text{ s.t. } \sum_{j=1}^{A(t)} P(s_j(t)) \leq N, \tag{1}$$

where $n(t)$ is the number of outstanding jobs at time t, and $A(t)$ is the set of jobs that are being processed at time t.

III. Metric

We represent the optimal offline algorithm (that knows the entire job arrival sequence including the number of phases, and the respective sizes of jobs in each phase, in advance) as OPT. Let $n(t)$ ($n_o(t)$) be the number of outstanding jobs with an online algorithm $\mathcal{A}$ (OPT). For Problem (1), we will consider the metric of competitive ratio which for an online algorithm $\mathcal{A}$ is defined as

$$\mu_{\mathcal{A}} = \max_{\sigma} \frac{\int n(t)dt}{\int n_o(t)dt}, \tag{2}$$

where σ is the input sequence consisting of jobs set $\mathcal{J}$.

We will propose an online algorithm $\mathcal{A}$, and bound $\mu_{\mathcal{A}} \leq \kappa$, by showing that for each time instant t

$$n(t) + d\Phi(t)/dt \leq \kappa n_o(t), \tag{3}$$

where $\Phi(t)$ is some function called the **potential function** that satisfies the boundary conditions:

- $\Phi(t) = 0$ initially before all job arrivals and $\Phi(\infty) = 0$.
- $\Phi(t)$ does not increase on any job arrival or job departure with the algorithm or the OPT.

Integrating (3) over time, implies that the competitive ratio of $\mathcal{A}$ is at most κ.

IV. Algorithm Fractional-LCFS

In this section, we propose an algorithm that is semi non-clairvoyant, that disregards the information about the remaining job sizes of all the remaining phases, and only exploits the binary information about a job being in the elastic or the in-elastic phase, which will be compared against a clairvoyant optimal offline algorithm in terms of the competitive ratio. At time t, let the outstanding number of jobs in the system be $n(t)$, and $n_\iota(t)$ be the number of jobs that are in their in-elastic phase. Thus, $n(t) = n_e(t) + n_\iota(t)$, where $n_e(t)$ is the number of jobs that are in their elastic phase.

Scheduling and speed selection: Let β, θ be constants with $0 < \theta < \beta < 1$.

Case I $\frac{N}{\beta n(t)} \leq 1$: Process the $\beta n(t)$ jobs that have arrived **most recently** without distinguishing between jobs that are in their elastic or in-elastic phase. [1] **Speed:** Each of the $\beta n(t)$ jobs are processed at equal speed

$$s(t) = P\left(\frac{N}{\beta n(t)}\right). \tag{4}$$

[1]If $\beta n(t)$ is fractional, then we mean $\lceil \beta n(t) \rceil$.

Case II $\frac{N}{\beta n(t)} > 1$: IIa: If $n_\iota(t) \geq \theta n(t)$[2] then process any $\min\{n_\iota(t), N\}$ jobs[3] that are in their in-elastic phase, and among the $n_e(t)$ jobs that are in their elastic phase, process the $\beta n_e(t)$ that have arrived **most recently**. **Speed:** $s(t)$

$$= \begin{cases} 1 & \text{for each of} \min\{n_\iota(t), N\} \text{ jobs,} \\ P\left(\frac{N-\min\{n_\iota(t),N\}}{\beta n_e(t)}\right) & \text{for each of } \beta n_e(t) \text{ jobs.} \end{cases} \tag{5}$$

IIb:If $n_\iota(t) < \theta n(t)$ Among the $\beta n(t)$ jobs that have arrived **most recently**, process all the jobs that are in their elastic phases with equal speed

$$s(t) = P\left(\frac{N}{\beta n(t)}\right). \tag{6}$$

Note that in this subcase, the total speed constraint of $\sum_{j=1}^{A(t)} P^{-1}(s_j(t)) \leq N$ need not be tight. Thus, for a practical implementation, few more jobs can be processed, however, that will not change the analysis.

By its very definition, algorithm Fractional-LCFS satisfies the total speed constraint of $\sum_{j=1}^{A(t)} P^{-1}(s_j(t)) \leq N$, as well as the speed constraint of unity for any job that is in its in-elastic phase.

The main result of this paper is as follows.

Theorem 1. *For any $\alpha > 1$, there exists a $0 < \theta < \beta < 1$, such that the competitive ratio of algorithm* Fractional-LCFS *for Problem* (1) *is a constant (depends only on α) and is independent of the number of jobs, their sizes, and the number of servers N. The exact competitive ratio expression is provided in* (22)*, and using which for example in case of $\alpha = 2$, we get the competitive ratio bound of* 636*, choosing $\beta = \frac{1}{6}$, and $\theta = \frac{1}{72}$.*

For each value of $\alpha > 1$, how to choose β, θ such that the competitive ratio remains a constant is discussed in Remark 4. Note that in Remark 4 we prescribe only one possible choice of parameters θ, β that is sufficient to make the competitive ratio as a constant. However, there is scope for choosing the parameters θ, β so as to minimize the competitive ratio, which is analytically challenging, but numerically easy.

Remark 2. *It is worth noting that the competitive ratio bound in Theorem 1 increases as $\alpha \to 1$. The main intuition for this is that we are considering the worst case input, which includes the case where jobs have no in-elastic phases, for which as $\alpha \to 1$, SRPT is an optimal algorithm that processes only one job with the least remaining size on all servers. In contrast, with* Fractional-LCFS*, potentially a large number of jobs are parallely processed with equal speed for all values of α.*

Discussion: Even though the derived competitive ratio of Fractional-LCFS appears large, it overcomes an old technical hurdle of it being independent of system parameters. In prior work, either speed augmentation [4] was shown to be necessary to get similar constant competitive ratio results, or somewhat simplistic input model had to be considered [15].

[2]If $\theta n(t)$ is fractional, then we mean $\lceil \theta n(t) \rceil$.
[3]If $N = \min\{n_\iota(t), N\}$, then pick any N jobs out of total $n_\iota(t)$ jobs.

Moreover, given the very nature of the competitive ratio metric being a multiplicative penalty, a large competitive ratio per se is not limiting, as long as it does not scale with system parameters.

The intuition as to why a fractional LCFS algorithm should perform well is similar to that of the SRPT (shortest remaining processing time) algorithm that requires the knowledge of remaining job sizes. SRPT minimizes the number of outstanding jobs (that controls the flow time) knowing the jobs sizes, by keeping shorter jobs in the system for less time. Fractional LCFS on the other hand, without using the remaining job size information, processes a fraction of the most recently arrived jobs, and tries to keep longer jobs stay in the system for long, thus 'effectively' prioritizing short jobs. It is easy to construct 'bad' input sequences where this is not the case, but roughly that is what one should expect.

Moreover, the intuition for the equal speed choice can be borrowed from [4], that explains that if an algorithm choosing equal speed has more number of outstanding jobs than the OPT, then progressively, it allocates fewer servers to each job and since $\alpha > 1$, it improves the utilization of servers. Since we also have jobs that are in their in-elastic phase, this is not precisely correct, however, provides partial explanation.

After dealing with the setting where jobs arrive at arbitrary times, next, we consider the simpler case when all jobs are available at time 0 and get a better competitive ratio guarantee.

V. All jobs available at time 0

In this section, except for all jobs arriving at time 0, everything is identical to the system model described in Section II. For this case, we propose an algorithm PA-EQUI that makes the following choice for scheduling and speed selection.

Case I $\frac{N}{n(t)} \le 1$: Process all the $n(t)$ (number of outstanding) jobs, without distinguishing between jobs that are in their elastic or in-elastic phase, with equal speed $s(t) = P\left(\frac{N}{n(t)}\right)$.

Case II $\frac{N}{n(t)} > 1$: IIa: For a constant $0 < \delta < 1$, if $n_\iota(t) \ge \delta n(t)$ then process all $n_\iota(t)$ jobs that are in their in-elastic phase dedicatedly in one server with unit speed, while process the remaining $n_e(t)$ jobs that are in their elastic phase, each with speed $P\left(\frac{N-n_\iota(t)}{n_e(t)}\right)$.

IIb:If $n_\iota(t) < \delta n(t)$ Process all the $n_e(t)$ jobs that are in their elastic phase, each with equal speed $s(t) = P\left(\frac{N}{n_e(t)}\right)$. We name this algorithm PA-EQUI, since it allocates equal speed to all jobs that belong to the same phase. In contrast, EQUI studied in [4], [5] is Blind-EQUI, since it is unaware which jobs belong to which phase, and wastes speed. By its very definition, algorithm PA-EQUI satisfies the total speed constraint of $\sum_{j=1}^{A(t)} P^{-1}(s_j(t)) \le N$, as well as the speed constraint of unity for any job that is in its in-elastic phase.

The main result of this section is as follows.

Theorem 3. *For $\alpha > 1$, the competitive ratio of* PA-EQUI *for Problem* (1) *when all jobs are available at time* 0*, is at most*

$$\mu(\alpha) = \frac{1}{\alpha(1-\delta)-1}\left[\frac{\alpha(1-\delta)}{\delta} + \frac{\alpha(1-\delta)+\delta}{1-\delta}\right],$$

where δ is the parameter to be chosen. For $\alpha = 2$, choosing $\delta = \frac{1}{4}, \mu(2) = 50/3$. Moreover, $\mu(\alpha)$ is a decreasing function of $\alpha > 1$ for an appropriate choice of δ.

Proof is similar to that of Theorem 1 and for lack of space is omitted. It can be found in the full version [22]. Thus, compared to the online job arrivals case (Theorem 1) where the competitive ratio for $\alpha = 2$ is 636, there is a significant improvement in the competitive ratio when all jobs are available at time 0. Similar conclusion can be drawn for other values of α also.

VI. Numerical results

In this section, we present simulation results for the mean flow time (per job). We compare the performance of the proposed algorithm Fractional-LCFS with other known algorithms such as inelastic first IF [15], EQUI [4] and phase-aware FCFS PA-FCFS [19]. With PA-FCFS, jobs are processed in the order in which they arrive, and the earliest arrived job is processed by as many servers as possible, i.e. if a job is in its inelastic phase then one server is allocated and other jobs are considered similarly over the remaining number of servers, while if a job is in its elastic phase then all the available servers are allocated to that.

For all simulations, we use $\alpha = 2$. In Fig. 1, we let the number of servers to be $N = 10$, and consider a slotted time system, and plot the per-job flow time as a function of the per-slot mean arrival rate arr, where in each slot, the number of jobs arriving is Poisson distributed with the respective arr. For each job, the first/last phase is equally likely to be an elastic/in-elastic phase, and the number of phases of each job is Poisson distributed with mean 7. The choice of 7 is dictated by real-world datasets [10]. For each phase, each job's size is exponentially distributed with mean 5. For each iteration, we generate jobs for 1000 slots, and count its flow time, and iterate over 1000 iterations. For Fractional-LCFS, we choose $\theta = \frac{1}{4}$. We compare the performance of different algorithms for the same realization of random variables, and then average it out. More results for different values of number of servers can be found in the full version [22].

As we see from Fig. 1, the performance of Fractional-LCFS is similar to the inelastic first IF and the EQUI [4] algorithm, however, the mean flow time of the PA-FCFS is approximately 2 or 3 times larger than that of the other algorithms. With $\alpha = 2$, the limitation of PA-FCFS is that whenever the earliest arrived job in its elastic phase, the speed dedicated to it is $N^{1/2}$ and no other job is processed. All the other three algorithms, in contrast, process multiple jobs with total speed roughly equal to $n(N/n)^{1/2}$ (n is the number of outstanding jobs), thus having a far better performance.

In Fig. 2, we plot the performance of Fractional-LCFS for different choices of β with $\theta = 1/4$ and mean per-slot arrival rate of 10, and the rest of settings are the same as in Fig. 1. In the theoretical result we showed that for $\alpha = 2$ with $\beta = 1/6$ and $\theta = 1/72$, the competitive ratio of Fractional-LCFS is a constant. From Fig. 2 we observe that in fact the performance of FRACTIONAL-LCFS improves by choosing larger values of β and θ, and the choice of $\beta = 1/6$ and $\theta = 1/72$ was needed only for theoretical purposes. Fig. 2 shows that $\beta = 1$ has the best performance among different choices of β for FRACTIONAL-LCFS.

Simulation results for the case when all jobs are available at time 0 can be found in the full version [22].

VII. CONCLUSIONS

In this paper, we considered an important problem of flow time minimization in data centers, where jobs migrate between two phases of parallelizability (called elastic and in-elastic) multiple times. In the elastic phase, there is flexibility of parallelizing the job over multiple servers, while in the in-elastic phase, the job has to be processed by a single server. Moreover, in the elastic phase there is limited parallelizability, and the speed increment diminishes as more and more servers are allocated to any job. We considered the online setting, where jobs arrive over time with arbitrary sizes and arrival times, and proposed a LCFS type algorithm for scheduling, that processes the scheduled jobs with equal speed. We showed that its competitive ratio is a constant that only depends on the speed-up exponent α as long as $\alpha > 1$. With arbitrary input, our result overcomes fundamental difficulty found in literature where similar results were shown only in the presence of resource augmentation or for simpler model, by exploiting the specific structure of the problem with just two phases that is practically well motivated.

REFERENCES

[1] B. Berg, J.-P. Dorsman, and M. Harchol-Balter, "Towards optimality in parallel scheduling," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–30, 2017.

[2] B. Berg, R. Vesilo, and M. Harchol-Balter, "heSRPT: Optimal scheduling of parallel jobs with known sizes," *SIGMETRICS Perform. Evaluation Rev.*, vol. 47, no. 2, pp. 18–20, 2019. [Online]. Available: https://doi.org/10.1145/3374888.3374896

[3] S. Im, B. Moseley, K. Pruhs, and E. Torng, "Competitively scheduling tasks with intermediate parallelizability," *ACM Transactions on Parallel Computing (TOPC)*, vol. 3, no. 1, pp. 1–19, 2016.

[4] J. Edmonds, "Scheduling in the dark," *Theoretical Computer Science*, vol. 235, no. 1, pp. 109–141, 2000.

[5] J. Edmonds and K. Pruhs, "Scalably scheduling processes with arbitrary speedup curves," in *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 2009, pp. 685–692.

[6] K. Agrawal, J. Li, K. Lu, and B. Moseley, "Scheduling parallelizable jobs online to minimize the maximum flow time," in *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*, 2016, pp. 195–205.

[7] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at google with borg," in *Proceedings of the Tenth European Conference on Computer Systems*, 2015, pp. 1–17.

[8] S.-H. Lin, M. Paolieri, C.-F. Chou, and L. Golubchik, "A model-based approach to streamlining distributed training for asynchronous sgd," in *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2018, pp. 306–318.

[9] R. Vaze and J. Nair, "Speed scaling with multiple servers under a sum power constraint," in *Performance 2021*, 2021.

[10] "Noisepage -the self-driving database management system." [Online]. Available: https://noise.page

[11] P. O'Neil, E. O'Neil, X. Chen, and S. Revilak, "The star schema benchmark and augmented fact table indexing," in *Technology Conference on Performance Evaluation and Benchmarking*. Springer, 2009, pp. 237–252.

[12] N. R. Tallent and J. M. Mellor-Crummey, "Effective performance measurement and analysis of multithreaded applications," in *Proceedings of the 14th ACM SIGPLAN symposium on Principles and practice of parallel programming*, 2009, pp. 229–240.

[13] T. D. Nguyen, R. Vaswani, and J. Zahorjan, "Using runtime measured workload characteristics in parallel processor scheduling," in *Workshop on Job Scheduling Strategies for Parallel Processing*. Springer, 1996, pp. 155–174.

[14] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*. Ieee, 2010, pp. 1–10.

[15] B. Berg, J. Whitehouse, B. Moseley, W. Wang, and M. Harchol-Balter, "The case for phase-aware scheduling of parallelizable jobs," *Performance Evaluation*, p. 102246, 2021.

[16] R. Ren and X. Tang, "Clairvoyant dynamic bin packing for job scheduling with minimum server usage time," in *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*, 2016, pp. 227–237.

[17] J. Turek, W. Ludwig, J. L. Wolf, L. Fleischer, P. Tiwari, J. Glasgow, U. Schwiegelshohn, and P. S. Yu, "Scheduling parallelizable tasks to minimize average response time," in *Proceedings of the sixth annual ACM symposium on Parallel algorithms and architectures*, 1994, pp. 200–209.

[18] J. Turek, U. Schwiegelshohn, J. L. Wolf, and P. S. Yu, "Scheduling parallel tasks to minimize average response time," in *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*, 1994, pp. 112–121.

[19] V. Leis, P. Boncz, A. Kemper, and T. Neumann, "Morsel-driven parallelism: A numa-aware query evaluation framework for the many-core age," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 743–754. [Online]. Available: https://doi.org/10.1145/2588555.2610507

[20] B. Berg, M. Harchol-Balter, B. Moseley, W. Wang, and J. Whitehouse, "Optimal resource allocation for elastic and inelastic jobs," in *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures*, 2020, pp. 75–87.

[21] R. Motwani, S. Phillips, and E. Torng, "Nonclairvoyant scheduling," *Theoretical computer science*, vol. 130, no. 1, pp. 17–47, 1994.

[22] R. Vaze, "Scheduling for multi-phase parallelizable jobs," 2022. [Online]. Available: https://arxiv.org/abs/2205.00518

APPENDIX A

A. *Proof of Theorem 1*

From here on we refer to algorithm FRACTIONAL-LCFS as just algorithm. Let at time t, the set of outstanding (unfinished) number of jobs with the algorithm be $A(t)$ with $n(t) = |A(t)|$. Similarly, let $O(t)$ be the set of outstanding jobs with the OPT at time t. Let at time t, the **rank** $r_j(t)$ of a job $j \in A(t)$ be equal to the number of outstanding jobs of $A(t)$ with the algorithm that have arrived before job j. Note that the rank of a job does not change on arrival of a new job, but can change if a job departs that had arrived earlier.

Let $Q(x) = \frac{x}{P(x)}$. which specializes to $Q(x) = x^{1-\frac{1}{\alpha}}$ for $P(x) = x^{1/\alpha}$. Moreover, let $\{x\}^+ = \max\{x, 0\}$. Then we consider the following potential function

$$\Phi(t) = c_1\Phi_1(t) + c_2\Phi_2(t), \tag{7}$$

where

$$\Phi_1(t) = \sum_{j \in A(t)} \frac{r_j(t)}{P(N)Q(r_j(t))} \left(w_j^A(t) - w_j^o(t)\right)^+, \tag{8}$$

and

$$\Phi_2(t) = \sum_{j \in A(t)} \bar{w}_{j\iota}^A(t) - \sum_{j \in O(t)} \bar{w}_{j\iota}^o(t), \tag{9}$$

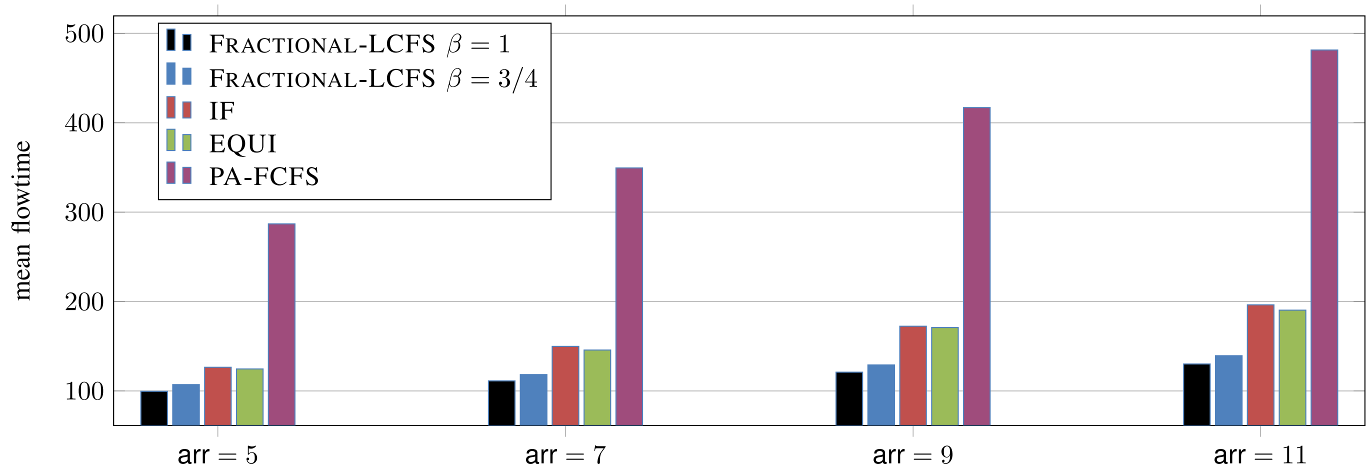

Fig. 1. Comparison of mean flow time with different algorithms as a function of mean arrival rate per slot with 10 servers.

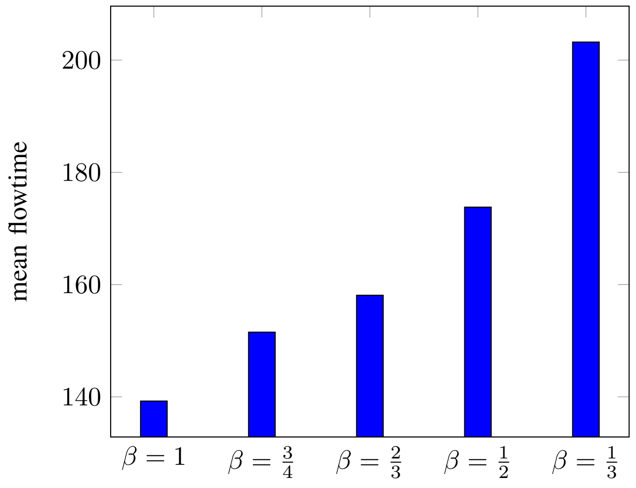

Fig. 2. Comparison of flow time for FRACTIONAL-LCFS with different choices of β for $\theta = 1/4$ with mean per-slot arrival rate of 10 with 10 servers.

where $w_j^A(t)$ ($w_j^o(t)$) is the remaining size (sum of the job sizes of all the remaining elastic and in-elastic phases) of job j with the algorithm (OPT) at time t, while $\bar{w}_{j\iota}^A(t)$ ($\bar{w}_{j\iota}^o(t)$) is the sum of the remaining size of job j in all its remaining in-elastic phases with the algorithm (OPT) at time t, and c_1, c_2 are constants to be chosen later.

We next show that the potential function $\Phi(t)$ satisfies the second boundary condition. The fact that the first boundary condition is satisfied is trivial.

Lemma 1. *Potential function* $\Phi(t)$ (7) *does not change on arrival of any new job. Moreover, on a departure of a job with the algorithm or the* OPT, *the potential function* $\Phi(t)$ (7) *does not increase.*

The proofs of Lemma 1 and 2 are provided in Appendix B.

We next bound the drift $d\Phi(t)/dt$ because of the processing by the OPT, and the algorithm, respectively. To avoid cumbersome notation, we write $\beta n(t)$ or $\theta n(t)$ instead of $\lceil \beta n(t) \rceil$ or $\lceil \theta n(t) \rceil$everywhere.

Lemma 2. *The change in the potential function* (7) *because of the* $\mathsf{OPT}'s$ *contribution*

$$d\Phi(t)/dt \leq c_1 n(t) \frac{Q(n_o(t))}{Q(n(t))} + c_2 n_o(t). \qquad (10)$$

Lemma 3. *With* $0 < \theta + \gamma < \beta$, *for any* t *where* $n_o(t) \leq \gamma n(t)$, *the change in the potential function* (7) *because of the algorithm's contribution is* $d\Phi(t)/dt$

$$\leq \begin{cases} -c_1 \frac{(1-\beta)(\beta-\gamma)n(t)}{P(\beta)} & \text{if } \frac{N}{\beta n(t)} \leq 1, \\ -c_2 \min\{N, n_\iota(t)\} & \text{if } \frac{N}{\beta n(t)} > 1 \text{ and } n_\iota(t) \geq \theta n(t), \\ -c_1 \frac{(1-\beta)(\beta-\theta-\gamma)n(t)}{P(\beta)}, & \text{otherwise}. \end{cases} \qquad (11)$$

The proof of Lemma 3 is provided in Appendix C. To prove Theorem 1, we check the running condition (3) for the following two cases separately for a fixed γ such that $\theta + \gamma < \beta$ (choice to be made later) : i) $n_o(t) > \gamma n(t)$ and ii) $n_o(t) \leq \gamma n(t)$, and show that it holds for a constant κ.

Case i) $n_o(t) > \gamma n(t)$. In this case, we only count the $\mathsf{OPT}'s$ contribution to $d\Phi(t)/dt$, which is sufficient since the algorithm's contribution to $d\Phi(t)/dt$ is always non-positive. From Lemma 2, we have that

$$\begin{aligned} n(t) + d\Phi(t)/dt &\leq n(t) + c_1 n(t)\frac{Q(n_o(t))}{Q(n(t))} + c_2 n_o(t), \\ &\overset{(a)}{\leq} n(t) + c_1 n(t)\frac{Q(bn(t))}{Q(n(t))} + c_2 n_o(t), \\ &= n(t) + c_1 n(t) b^{1-1/\alpha} + c_2 n_o(t), \\ &\overset{(b)}{\leq} n(t) + c_1 n(t) b + c_2 n_o(t), \\ &= n(t) + c_1 n_o(t) + c_2 n_o(t), \qquad (12) \\ &\overset{(c)}{\leq} (1/\gamma + c_1 + c_2) n_o(t), \qquad (13) \end{aligned}$$

where in (a) we let $n_o(t) = bn(t)$ and inequality (b) follows when $b > 1$. Finally (c) follows since $n_o(t) > \gamma n(t)$. When $b < 1$, then $\frac{Q(bn(t))}{Q(n(t))} < 1$. Thus, similar to (13), for $b < 1$, we get

$$n(t) + d\Phi(t)/dt \le n(t) + c_1 n(t)\frac{Q(n_o(t))}{Q(n(t))} + c_2 n_o(t),$$
$$\le n(t)(1+c_1) + c_2 n_o(t),$$
$$\le \left(\frac{1+c_1}{\gamma} + c_2\right) n_o(t). \tag{14}$$

Case ii) $n_o(t) \le \gamma n(t)$. Let $n_o(t) > 0$.

ii-a) With $\frac{N}{\beta n(t)} \le 1$, from Lemma 2 and Lemma 3, (3) can be bounded as $n(t) + d\Phi(t)/dt$

$$\le n(t) + c_1 n(t)\frac{Q(n_o(t))}{Q(n(t))} + c_2 n_o(t) - c_1 \frac{(1-\beta)(\beta-\gamma)}{P(\beta)} n(t),$$
$$\overset{(a)}{\le} c_2 n_o(t) + n(t)\left(1 + c_1\left(\gamma^{1-1/\alpha} - \frac{(1-\beta)(\beta-\gamma)}{P(\beta)}\right)\right),$$
$$\overset{(b)}{\le} c_2 n_o(t), \tag{15}$$

where (a) follows since $n_o(t) \le \gamma n(t)$, while (b) follows for choice of γ, β, c that satisfy

$$\frac{(1-\beta)(\beta-\gamma)}{P(\beta)} > \gamma^{1-1/\alpha} \text{ and } c_1 \ge \frac{-1}{\left(\gamma^{1-1/\alpha} - \frac{(1-\beta)(\beta-\gamma)}{P(\beta)}\right)}. \tag{16}$$

ii-b) When $\frac{N}{\beta n(t)} > 1$ and $n_\iota(t) \ge \theta n(t)$, from Lemma 2 and Lemma 3, (3) can be bounded as $n(t) + d\Phi(t)/dt$

$$\le n(t) + c_1 n(t)\frac{Q(n_o(t))}{Q(n(t))} + c_2 n_o(t) - c_2 \min\{N, n_\iota(t)\},$$
$$\overset{(a)}{\le} c_2 n_o(t) + n(t)\left(1 + c_1\gamma^{1-1/\alpha} - c_2\theta\right),$$
$$\overset{(b)}{\le} c_2 n_o(t), \tag{17}$$

where (a) follows since $n_o(t) \le \gamma n(t)$, $n_\iota(t) \ge \theta n(t)$, $\frac{N}{\beta n(t)} > 1$ and $\theta < \beta$, while (b) follows for

$$c_2 \ge \frac{(1 + c_1\gamma^{1-1/\alpha})}{\theta}. \tag{18}$$

ii-c) Finally, when $\frac{N}{\beta n(t)} > 1$ and $n_\iota(t) < \theta n(t)$, from Lemma 2 and Lemma 3, (3) can be bounded as $n(t)+d\Phi(t)/dt$

$$\le n(t) + \frac{c_1 n(t) Q(n_o(t))}{Q(n(t))} - \frac{c_1(1-\beta)(\beta-\theta-\gamma)n(t)}{P(\beta)}$$
$$+ c_2 n_o(t),$$
$$\overset{(a)}{\le} c_2 n_o(t) + n(t)\left(1 + c_1\left(\gamma^{1-1/\alpha} - \frac{(1-\beta)(\beta-\theta-\gamma)}{P(\beta)}\right)\right),$$
$$\overset{(b)}{\le} c_2 n_o(t), \tag{19}$$

where (a) follows since $n_o(t) \le \gamma n(t)$, while (b) follows for choice of γ, β, c that satisfy

$$\frac{(1-\beta)(\beta-\theta-\gamma)}{P(\beta)} > \gamma^{1-1/\alpha} \tag{20}$$

and
$$c_1 \ge \frac{-1}{\left(\gamma^{1-1/\alpha} - \frac{(1-\beta)(\beta-\theta-\gamma)}{P(\beta)}\right)}. \tag{21}$$

When $n_o(t) = 0$, the OPT's contribution to $d\Phi(t)/dt$ is zero, and we can bound (3) with smaller value of κ. Combining (15), (17), (19), together with (13) and (14), the competitive ratio of the proposed algorithm is at most

$$\frac{1+c_1}{\gamma} + c_2 \tag{22}$$

for β, θ, γ, that satisfy (16), (18), (20) and (21). Depending on $\alpha > 1$, there exists a $\beta < 1$ satisfying (16), (20) and (21) with $\theta = \gamma = \beta^2/2$ as follows. In particular, with $\theta = \gamma = \beta^2/2$, to satisfy (16), (20) and (21) i.e., $\frac{(1-\beta)(\beta-\gamma)}{P(\beta)} > \gamma^{1-1/\alpha}$ and $\frac{(1-\beta)(\beta-\theta-\gamma)}{P(\beta)} > \gamma^{1-1/\alpha}$, it is sufficient that $1 - 2\beta + \beta^2 > \frac{\beta}{2}^{1-1/\alpha}$. Since $\alpha > 1$, $1 - 2\beta + \beta^2 - \left(\frac{\beta}{2}\right)^{1-1/\alpha} = 1$ at $\beta = 0$. Thus, using continuity, we know that there exists a $0 < \beta < 1$ satisfying (16), (20) and (21) with $\theta = \gamma = \beta^2/2$. This implies that the competitive ratio is a constant that only depends on α and not on any other system parameter. Moreover, notice that as $\alpha \to 1$, the appropriate choice of β decreases implying that the competitive ratio (22) increases.

For example, for $\alpha = 2$, let $\beta = \frac{1}{6}$ and $\theta = \gamma = \beta^2/2$, $c_1 = \frac{-1}{\left(\gamma^{1-1/\alpha} - \frac{(1-\beta)(\beta-\theta-\gamma)}{P(\beta)}\right)} = 6.06$, $c_2 = \frac{(1+c_1\gamma^{1-1/\alpha})}{\theta} = 72(1 + .77) = 127.44$. We get a competitive ratio of $\frac{1+c_1}{\gamma} + c_2 \le 72 \times (1 + 6.06) + 127.44 = 635.76$.

Analytically optimizing the competitive ratio with respect to the variables, β, α, and γ could result in a much lower bound, however, appears difficult. Numerically, however, one can easily do so.

Remark 4. *For any $\alpha > 1$, choosing $\theta = \gamma = \beta^2/2$, and $0 < \beta < 1$ such that $1 - 2\beta + \beta^2 > \frac{\beta}{2}^{1-1/\alpha}$ is sufficient to make the competitive ratio constant. Moreover, finding such a β is easy numerically.*

APPENDIX B

Proof of Lemma 1. On an arrival of a new job j, the ranks of all the existing jobs do not change, while for the newly arrived job j, $w_j^A(t) - w_j^o(t) = 0$. Hence the potential function $\Phi_1(t)$ (7) does not change on arrival of any new job.

On a departure of a job with the algorithm, rank of any remaining job can only decrease, in particular by 1. Thus, if at time t when job k departs with the algorithm, job j's ($j \in A(t^+)$) rank at time t^+, is either $r_j(t^+) = r_j(t)$ or $r_j(t^+) = r_j(t) - 1$. Since function $\frac{r_j(t)}{Q(r_j(t))}$ is a non-decreasing function, thus the potential function $\Phi_1(t)$ does not increase on departure of a job with the algorithm.

For the OPT, only $w_j^o(t)$ decreases with job processing and that too smoothly. Thus, there is no discontinuity when a job departs with the OPT, hence $\Phi_1(t)$ does not change when a job departs with the OPT. Moreover, for $\Phi_2(t)$, on an arrival of a new job $\bar{w}_{j\iota}^A(t) - \bar{w}_{j\iota}^o(t) = 0$, while there is no discontinuity when a job departs with the OPT or the algorithm, since both $\bar{w}_{j\iota}^A(t)$ and $\bar{w}_{j\iota}^o(t)$ decrease smoothly. Hence $\Phi_2(t)$ does not change when a new job arrives or a job departs with the OPT or the algorithm. □

Proof of Lemma 2. We begin with the following simple result whose proof is immediate.

Lemma 4. *Disregarding the unit speed constraint for any job whose in-elastic part is being processed, the maximum speed devoted to processing any one job by the* OPT *is at most*

$P(N)$. Moreover, the sum of the speeds with which OPT *is processing any of its k jobs is at most $Q(k)P(N)$.*

From the definition of $\Phi(t)$ (7), OPT can increase $\Phi_1(t)$ at time t only if it processes jobs that also belong to the set $A(t)$ (outstanding jobs with the algorithm). Thus, from Lemma 4, the maximum sum of the speeds devoted to the set of $A(t)$ jobs by the OPT is at most $Q(n(t))P(N)$, where each job gets processed at speed $P\left(\frac{N}{n(t)}\right)$. Moreover, by definition, OPT contains only $n_o(t)$ jobs. Thus, the sum of the speeds devoted to the $n(t)$ jobs of the algorithm by the OPT is at most $Q(\min\{n(t), n_o(t)\})P(N)$. From the definition of $\Phi_1(t)$ (8), the maximum increase in $\Phi_1(t)$ is possible if the total speed of the OPT that it can dedicate to jobs belonging to $A(t)$ is dedicated to the single job with the largest rank among $A(t)$, i.e., the job with rank equal to $n(t)$. Thus, because of processing by the OPT

$$\begin{aligned} d\Phi_1(t)/dt &\leq \frac{n(t)}{P(N)Q(n(t))} \times Q(\min\{n(t), n_o(t)\})P(N), \\ &\leq n(t)\frac{Q(n_o(t))}{Q(n(t))}. \end{aligned} \tag{23}$$

Moreover, any job that is in its in-elastic phase can be processed with at most unit speed. Since there are at most $n_o(t)$ jobs with the OPT that are in their in-elastic phases, we get

$$d\Phi_2/dt \leq n_o(t).$$

□

Appendix C

Proof of Lemma 3. Case I : $\left(\frac{N}{\beta n(t)}\right) \leq 1$ Since the algorithm processes the $\beta n(t)$ jobs that have arrived most recently, the rank of job i that is being processed by the algorithm is $r_i(t) = n(t) - i + 1$ for $i = 1, \ldots, \beta n(t)$. Since $n_o(t) \leq \gamma n(t)$, and $\gamma < \beta$, $w_j^A(t) - w_j^o(t) > 0$ for at least $(\beta - \gamma)n(t)$ jobs with the algorithm. In the worst case, the ranks of these $(\beta-\gamma)n(t)$ jobs are $(1-\beta)n(t) + i - 1$ for $i = 1, \ldots, (\beta - \gamma)n(t)$.

Since the speed for any of the job processed by the algorithm is $s(t) = P\left(\frac{N}{\beta n(t)}\right)$, the change in the potential function because of the algorithm's processing to $\Phi_1(t)$ is

$$\begin{aligned} d\Phi_1(t)/dt &\leq -\sum_{i=(1-\beta)n(t)}^{(1-\beta)n(t)+(\beta-\gamma)n(t)} \frac{r_i(t)}{P(N)Q(r_i(t))} P\left(\frac{N}{\beta n(t)}\right), \\ &\overset{(a)}{\leq} -\sum_{i=(1-\beta)n(t)}^{(1-\beta)n(t)+(\beta-\gamma)n(t)} \frac{r_i(t)}{Q(n(t))}\frac{1}{P(\beta n(t))}, \\ &= -\sum_{i=(1-\beta)n(t)}^{(1-\beta)n(t)+(\beta-\gamma)n(t)} \frac{r_i(t)}{Q(n(t))}\frac{1}{P(n(t))}\frac{1}{P(\beta)}, \\ &\overset{(b)}{=} -\sum_{i=(1-\beta)n(t)}^{(1-\beta)n(t)+(\beta-\gamma)n(t)} \frac{r_i(t)}{n(t)}\frac{1}{P(\beta)}, \\ &\overset{(c)}{\leq} -\frac{(1-\beta)(\beta-\gamma)n(t)n(t)}{\beta}\frac{1}{n(t)P(\beta)}, \\ &= -\frac{(1-\beta)(\beta-\gamma)n(t)}{P(\beta)}, \end{aligned}$$

where (a) follows since $r_i(t) \leq n(t)$ and

$$\frac{P\left(\frac{N}{\beta n(t)}\right)}{P(N)} \geq \frac{1}{P(\beta n(t))},$$

while (b) follows since $Q(x)P(x) = x$, and finally (c) follows since there are $(\beta-\gamma)n(t)$ jobs that are being processed each with rank at least $(1-\beta)n(t)$. For $\Phi_2(t)$, in this case, we just bound $d\Phi_2(t)/dt \leq 0$ because of the algorithm's processing.

Case II : $\left(\frac{N}{\beta n(t)}\right) > 1$

IIa: $n_\iota(t) \geq \theta n(t)$ In this case, for the algorithm we will only consider the drift $d\Phi_2(t)/dt$, and trivially upper bound $d\Phi_1(t)/dt \leq 0$. When $n_\iota(t) \geq \theta n(t)$, each of the $\min\{N, n_\iota(t)\}$ jobs are processed at unit speed by the algorithm, and we get

$$d\Phi_2(t)/dt \leq -\min\{N, n_\iota(t)\}. \tag{24}$$

IIb: $n_\iota(t) < \theta n(t)$ In this case, for the algorithm we will only consider the drift $d\Phi_1(t)/dt$ and upper bound $d\Phi_2(t)/dt \leq 0$.

In this case, the algorithm processes those jobs that are in their elastic phases among the $\beta n(t)$ jobs that have arrived most recently. Since $n_\iota(t) < \theta n(t)$, and $n_\iota(t) + n_e(t) = n(t)$, at least $(\beta - \theta)n(t)$ jobs (that are in their elastic phases) are being processed.

Moreover, since $n_o(t) \leq \gamma n(t)$, for at least $(\beta - \theta - \gamma)n(t)$ jobs that are being processed by the algorithm $w_j^A(t) - w_j^o(t) > 0$, and the rank of each of these $(\beta - \theta - \gamma)n(t)$ jobs is at least $(1-\beta)n(t)$.

Since the speed for any of the job processed by the algorithm is $s(t) = P\left(\frac{N}{\beta n(t)}\right)$, the change in the potential function $\Phi_1(t)$ because of the algorithm's processing is

$$\begin{aligned} d\Phi_1(t)/dt &\leq -\sum_{i=1}^{(\beta-\theta-\gamma)n(t)} \frac{r_i(t)}{P(N)Q(r_i(t))} P\left(\frac{N}{\beta n(t)}\right), \\ &\overset{(a)}{\leq} -\sum_{i=1}^{(\beta-\theta-\gamma)n(t)} \frac{r_i(t)}{Q(n(t))}\frac{1}{P(\beta n(t))}, \\ &= -\sum_{i=1}^{(\beta-\theta-\gamma)n(t)} \frac{r_i(t)}{Q(n(t))}\frac{1}{P(n(t))}\frac{1}{P(\beta)}, \\ &\overset{(b)}{=} -\sum_{i=1}^{(\beta-\theta-\gamma)n(t)} \frac{r_i(t)}{n(t)}\frac{1}{P(\beta)}, \\ &\overset{(c)}{\leq} -\frac{(1-\beta)(\beta-\theta-\gamma)n(t)n(t)}{\beta}\frac{1}{n(t)P(\beta)}, \\ &= -\frac{(1-\beta)(\beta-\theta-\gamma)n(t)}{P(\beta)}, \end{aligned}$$

where (a) follows since $r_i(t) \leq n(t)$, while (b) follows since $Q(x)P(x) = x$, and finally (c) follows since there are $(\beta - \theta - \gamma)n(t)$ jobs that are being processed each with rank at least $(1-\beta)n(t)$.

□