Energy-Efficient Multiprocessor Scheduling for Flow Time and Makespan

Hongyang Sun* Yuxiong He† Wen-Jing Hsu‡ Rui Fan§

Abstract

We consider energy-efficient scheduling on multiprocessors, where the speed of each processor can be individually scaled, and a processor consumes power s^α when running at speed s, for $\alpha > 1$. A scheduling algorithm needs to decide at any time both processor allocations and processor speeds for a set of parallel jobs with time-varying parallelism. The objective is to minimize the sum of the total energy consumption and certain performance metric, which in this paper includes total flow time and makespan. For both objectives, we present instantaneous parallelism-clairvoyant (IP-clairvoyant) algorithms that are aware of the instantaneous parallelism of the jobs at any time but not their future characteristics, such as remaining parallelism and work. For total flow time plus energy, we present an $O(1)$-competitive algorithm, which significantly improves upon the best known non-clairvoyant algorithm and is the first constant competitive result on multiprocessor speed scaling for parallel jobs. In the case of makespan plus energy, which is considered for the first time in the literature, we present an $O(\ln^{1-1/\alpha} P)$-competitive algorithm, where P is the total number of processors. We show that this algorithm is asymptotically optimal by providing a matching lower bound. In addition, we also study non-clairvoyant scheduling for total flow time plus energy, and present an algorithm that achieves $O(\ln P)$-competitive for jobs with arbitrary release time and $O(\ln^{1/\alpha} P)$-competitive for jobs with identical release time. Finally, we prove an $\Omega(\ln^{1/\alpha} P)$ lower bound on the competitive ratio of any non-clairvoyant algorithm, matching the upper bound of our algorithm for jobs with identical release time.

Keywords: Multiprocessors, Online Scheduling, Dynamic speed scaling, Energy-performance tradeoff, Competitive Analysis, Total flow time, Makespan

1 Introduction

Energy has been widely recognized as a key consideration in the design of mobile and high-performance computing systems. One popular approach to controlling the energy consumption is by dynamically varying the speeds of the processors, a technique generally known as *dynamic speed scaling* [15, 26, 48]. Major chip manufacturers, such as Intel, AMD and IBM, have produced chips that enable the operating systems to perform dynamic power management using this technology. It has been observed that, for most CMOS-based processors, the dynamic power consumption satisfies the *cube-root* rule; that is, the power consumption of a processor is proportional to s^3 when it runs at speed s [15, 38]. Since the

*School of Computer Engineering, Nanyang Technological University, Singapore. sunh0007@ntu.edu.sg
†Microsoft Research, Redmond, WA, USA. yuxhe@microsoft.com
‡School of Computer Engineering, Nanyang Technological University, Singapore. hsu@ntu.edu.sg
§School of Computer Engineering, Nanyang Technological University, Singapore. fanrui@ntu.edu.sg

seminal paper by Yao, Demers and Shenker [49], who initiated the theoretical investigation of energy-efficient scheduling, most algorithmic researchers have assumed a more general power function of s^α, where $\alpha > 1$ is called the *power parameter*. As this power function is strictly convex, using dynamic speed scaling can result in a non-linear tradeoff between the energy consumption and the performance, and this has led to many interesting new research problems. One challenging problem concerns how to balance the conflicting objectives of low energy and high performance. The problem has attracted much attention among the algorithmic community and has become an active research topic in recent years. (See [2, 31] for two surveys of the field.)

In this paper, we study the challenging problem of scheduling parallel jobs on multiprocessors for the energy-performance tradeoff. We focus on systems with per-processor speed scaling capability; that is, the speed of each processor can be individually scaled [29, 50, 51]. This kind of architecture has been made possible by the recent advancements in chip design technology, such as the on-chip switching regulators [35, 34]. Under this setting, a scheduling algorithm needs to have both a *processor allocation policy*, which determines the number of processors allocated to each job, and a *speed scaling policy*, which determines the speed of each allocated processor. Moreover, we assume that the parallel jobs can have time-varying parallelism in different phases of their executions [22, 17, 44]. This poses an additional challenge compared to scheduling sequential jobs. In particular, it requires a scheduling algorithm to have dynamic policies in order to respond to the jobs' different resource requirements over time. If not designed properly, however, the algorithm could waste a large amount of energy or cause severe execution delays and hence performance degradations.

Our objective is to minimize a linear combination of energy consumption and certain performance metric, which in this paper includes total flow time and makespan. The *flow time* of a job is the duration between its release time and completion, and the *total flow time* is the sum of the flow time of all the jobs in the system. The *makespan* is the largest completion time of the jobs. Both total flow time and makespan are widely used performance metrics: The former measures the average response time of all users in the system, and the latter is closely related to the throughput of the system. Although energy and flow time (or makespan) have different units, optimizing a linear combination of the two has a natural interpretation if we consider a user who is willing to spend one unit of energy in order to reduce ρ units of total flow time (or makespan)[1]. In fact, minimizing the sum of conflicting objectives has been a common practice in many bi-criteria optimization problems [3, 36], and similar metrics have been considered previously in the scheduling literature that combine both performance and the cost of scheduling into a single objective function [47, 43, 20].

Since Albers and Fujiwara [3] first considered the problem of minimizing total flow time plus energy, many results (e.g., [7, 37, 36, 6, 16, 17, 44, 25, 4, 5]) have been obtained under different online scheduling settings. Some of these results assume that the scheduling algorithm is *clairvoyant*; that is, it gains complete knowledge of all job characteristics immediately upon the job's arrival. Other results are for an arguably more practical *non-clairvoyant* setting, where the scheduler knows nothing about the un-executed portion of a job. Most of these results, however, are only applicable to scheduling sequential jobs. Also, to the best of our knowledge, no previous work has considered minimizing makespan plus energy. The closest result to ours is by Chan, Edmonds and Pruhs [17], who studied

[1]By scaling the units of time and energy, we can assume without loss of generality that $\rho = 1$.

non-clairvoyant scheduling for parallel jobs on multiprocessors to minimize total flow time plus energy. In both [17] and our previous work [44], it has been observed that any non-clairvoyant algorithm that allocates a set of uniform-speed processors to a job will perform poorly; in particular, a lower bound of $\Omega(P^{(\alpha-1)/\alpha^2})$ on the competitiveness has been shown for any such algorithm, where P is the total number of processors. The reason is because a non-clairvoyant algorithm may in the worst case allocate a "wrong" number of processors to a job as compared to its parallelism, which will lead to either wasted energy or delayed job execution.

To obtain a better competitive ratio, it turns out that a non-clairvoyant algorithm needs to assign processors of different speeds to a job. To this end, Chan, Edmonds and Pruhs [17] proposed an execution model, in which a job can be simultaneously executed by several groups of processors. The processors within the same group must share the same speed, but different groups can run at different speeds. The execution rate of the job at any time is determined by the group with the fastest speed[2]. They proposed a non-clairvoyant algorithm called MultiLaps, and showed that it is $O(\log P)$-competitive with respect to total flow time plus energy for any set of parallel jobs. They also gave an $\Omega(\log^{1/\alpha} P)$ lower bound on the competitive ratio of any non-clairvoyant algorithm under this execution model.

In this paper, we first propose an alternative execution model, under which only one group of processors, possibly with different speeds, can be allocated to a job at any time. The execution rate of the job is determined by the speeds of the fastest processors that can be effectively utilized. This model is based on the assumption that the *maximum utilization policy* [32, 9] is employed at the underlying task scheduling level, which always utilizes faster processors before slower ones. Compared to the execution model proposed in [17], our model may be implemented more easily especially for data-parallel jobs with independent and sufficiently long tasks. Our first contribution includes a non-clairvoyant scheduling algorithm and its analysis under this execution model. The following states our results:

- We propose a non-clairvoyant algorithm N-EQUI (Non-uniform Equi-partitioning), and show that it is $O(\ln P)$-competitive with respect to the total flow time plus energy for any set of parallel jobs with arbitrary release time, and $O(\ln^{1/\alpha} P)$-competitive for jobs with identical release time. Moreover, we prove that any non-clairvoyant algorithm is $\Omega(\ln^{1/\alpha} P)$-competitive under our execution model, showing that N-EQUI is asymptotically optimal in the batch-released setting.[3]

Another contribution of this paper is to study a setting that lies between clairvoyance and non-clairvoyance. In this intermediate setting, a scheduling algorithm is allowed to know the available parallelism, or the *instantaneous parallelism (IP)*, of a job at any given time. The future characteristic of the job, such as its remaining parallelism or work, is still unknown. We call such an algorithm *IP-clairvoyant*[4]. In many parallel systems using centralized task queues or thread pools, instantaneous parallelism is simply the number of ready tasks in the queue or the number of ready threads in the pool, which is information practically available to the scheduler. Even for parallel systems using distributed scheduling

[2]In practice, this can be implemented by proper checkpointing of the executing program.

[3]It is interesting to observe that N-EQUI and MultiLaps achieve the same asymptotic competitive ratio under two different execution models that are not clearly related.

[4]This is to be distinguished from *semi-clairvoyant* scheduling [8], which is another intermediate setting that assumes a scheduling algorithm is able to gain approximate knowledge of a job upon its arrival, such as an estimate of its total work, but not the job's exact information.

Table 1: Competitive ratios of our non-clairvoyant and IP-clairvoyant algorithms for parallel jobs with arbitrary and identical release time for total response time plus energy.

	Non-clairvoyant	IP-clairvoyant
Arbitrary release time	$O(\ln P)$	$O(1)$
Identical release time	$\Theta(\ln^{1/\alpha} P)$	$2^{2-1/\alpha}+2$

such as work-stealing [10], instantaneous parallelism can be collected or estimated through counting or sampling without introducing much system overhead. It was shown previously that, when minimizing total flow time alone, knowledge about the instantaneous parallelism of the jobs provides limited benefit when compared to non-clairvoyant algorithms [21, 23, 33, 24]. However, we show in this paper that IP-clairvoyance can bring significant performance improvements when it comes to minimizing total flow time plus energy. Our contribution in this setting includes the following results:

- We present an IP-clairvoyant algorithm U-CEQ (Uniform Conservative Equi-partitioning), and show that it is $\left(\max\{\frac{4\alpha^2}{\alpha-1}, 4^\alpha\alpha\} + 2\alpha\right)$-competitive with respect to total flow time plus energy for any set of parallel jobs with arbitrary release time. This competitive ratio is independent of the total number P of processors, and therefore can be considered as constant for a fixed power parameter α. In addition, we show that U-CEQ is $(2^{2-1/\alpha}+2)$-competitive for any set of parallel jobs with identical release time.

Table 1 summarizes the competitive ratios of our algorithms under both non-clairvoyant and IP-clairvoyant settings. Compared to any non-clairvoyant algorithm, our IP-clairvoyant algorithm achieves significantly better competitive ratios, and in particular it gives the first constant competitive result on multiprocessor speed scaling for parallel jobs. The reason for the improvement comes from the fact that, given the instantaneous parallelism, an IP-clairvoyant algorithm can now allocate a "right" number of processors to a job at any time, ensuring that no energy will be wasted. At the same time, it can also guarantee a sufficient execution rate by setting the total power consumption proportionally to the number of active jobs at any time. This has been a common practice to designing online scheduling algorithms for total flow time plus energy and intuitively it provides the optimal balance between energy and performance [7, 37, 6, 16]. Moreover, unlike the non-clairvoyant algorithms MultiLaps and N-EQUI, both of which require non-uniform speed scaling for an individual job, U-CEQ only requires allocating processors of uniform speed to a job. Thus, in situations where the instantaneous parallelism of a job does not change frequently and can be effectively measured, e.g., by using feedback mechanisms [1, 28, 45], our IP-clairvoyant algorithm may be easier to implement and more practical.

Besides minimizing total flow time plus energy, there have been some recent studies that focus on the weighted variant of this problem [18], or optimize a linear combination of energy and some other performance metrics, such as total profit [40] and quoted lead time [19]. In this paper, we introduce a new objective function of minimizing makespan plus energy. Unlike the previous metrics, where the completion time of each job contributes to the overall objective function, makespan is only determined by the completion time of the last job in a job set, while the other jobs only contribute to the energy consumption part of the objective, and therefore can be slowed down to improve the overall performance. However, without knowing the future characteristics of the jobs, such as their remaining work, it is not clear even in the IP-clairvoyant setting which jobs should be slowed down in order to reduce energy without affecting the makespan. In the preliminary version [46] of this paper, we proposed an IP-clairvoyant algorithm that works for parallel jobs with identical release time and that consist of sequential phases and fully parallelizable phases up to all P processors. In this paper, we develop a generalized strategy that works for any set of parallel jobs regardless of their release time and parallelism structure. The following shows our contribution for minimizing makespan plus energy:

- We present an IP-clairvoyant algorithm WCEP (Work-Conserving Equal-Power) and show that it is $O(\ln^{1-1/\alpha} P)$-competitive with respect to makespan plus energy for any set of parallel jobs regardless of their release time, where P is the total number of processors. Moreover, we give a matching $\Omega(\ln^{1-1/\alpha} P)$ lower bound on the competitive ratio of any IP-clairvoyant algorithm, showing that WCEP is asymptotically optimal.

Finally, compared to minimizing total flow time plus energy, where a common strategy is to set the total power consumption at any time proportionally to the number of active jobs [7, 6, 16, 37], our results indicate that a good strategy for minimizing makespan plus energy is to set a constant power consumption at all time. In fact, both strategies share the same principle of balancing the costs incurred from both the power consumption and the target performance metric during the jobs' executions.

The rest of this paper is organized as follows. Section 2 formally defines the models and the objective functions. Section 3 presents our algorithms and analysis in both non-clairvoyant and IP-clairvoyant settings for the objective of total flow time plus energy. Section 4 presents our IP-clairvoyant algorithm for minimizing makespan plus energy. Finally, Section 5 concludes the paper with some discussions and future directions.

2 Models and Objective Functions

We consider a set $\mathcal{J} = \{J_1, J_2, \cdots, J_n\}$ of n jobs with time-varying parallelism to be scheduled on P processors whose speeds can be individually scaled. The power consumption of a processor running at speed s is given by s^α, where s can take any value in $[0, \infty)$ and $\alpha > 1$ is the *power parameter*. Adopting the notations used previously in [23, 22, 24, 17], each job $J_i \in \mathcal{J}$ contains k_i phases $\langle J_i^1, J_i^2, \cdots, J_i^{k_i}\rangle$, and each phase J_i^k is represented by an ordered pair $\langle w_i^k, h_i^k\rangle$, where $w_i^k \in \mathbb{R}^+$ denotes the amount of *work* and $h_i^k \in \mathbb{Z}^+$ denotes the *parallelism* of the phase[5]. Since a job can receive at most P processors at any time, it does not benefit by having a larger parallelism value than P. Hence, we can assume without loss of generality that $h_i^k \leq P$. A phase J_i^k is said to be *fully-parallelizable* if $h_i^k = P$ and it is *sequential* if $h_i^k = 1$. For convenience, we also define $x_i^k = w_i^k/(h_i^k)^{1-1/\alpha}$ to be the *unit-power span* for each phase J_i^k. This represents the time to complete the phase using exactly h_i^k processors of the same speed with a total power of 1 at all time. Suppose the h_i^k processors have the same speed s, we have $h_i^k s^\alpha = 1$, and so $s = (h_i^k)^{-\frac{1}{\alpha}}$. The amount of time to complete w_i^k amount of work is thus given by $w_i^k/(s \cdot h_i^k) = x_i^k$. For each job J_i, let $w(J_i) = \sum_{k=1}^{k_i} w_i^k$ denote its *total work* and let $x(J_i) = \sum_{k=1}^{k_i} x_i^k$ denote the job's *total unit-power span*.

[5]In [23, 22, 24, 17], an arbitrary non-decreasing and sub-linear speedup function is specified for each phase instead of a parallelism value, which represents a more general model for the jobs. For any non-clairvoyant algorithm, however, it was shown that the simple model used in this paper gives the hardest job instances for the combined objective of performance and energy [17].

Suppose that at some time t job J_i is in its k'th phase hence has parallelism h_i^k, and it is allocated $a_i(t)$ processors possibly with different speeds. Since a job cannot utilize more processors than its parallelism, its *effective processor allocation* at time t is given by $\bar{a}_i(t) = \min\{a_i(t), h_i^k\}$. The execution of the job is assumed to follow the *maximum utilization policy* [32, 9], which always utilizes faster processors before slower ones until all the allocated processors are utilized or the number of utilized processors reaches the parallelism of the job. In particular, let $s_{ij}(t)$ denote the speed of the j'th processor allocated to job J_i at time t, and we can assume without loss of generality that $s_{i1}(t) \geq s_{i2}(t) \geq \cdots \geq s_{ia_i(t)}(t)$. Then, only the $\bar{a}_i(t) = \min\{a_i(t), h_i^k\}$ fastest processors are utilized, and the *execution rate* $\Gamma_i^k(t)$ of the job is given by $\Gamma_i^k(t) = \sum_{j=1}^{\bar{a}_i(t)} s_{ij}(t)$. In the case where all the processors allocated to job J_i share the same speed $s_i(t)$, the execution rate is then simply $\Gamma_i^k(t) = \bar{a}_i(t)s_i(t)$.

At any time t, a scheduling algorithm needs to specify the number $a_i(t)$ of processors allocated to each job J_i, as well as the speed of each allocated processor. In this paper, we study two types of algorithms. An algorithm is said to be *non-clairvoyant* if it makes both scheduling decisions without any current or future information about a job, such as its release time, parallelism profile and remaining work. If an algorithm is aware of the current, or *instantaneous parallelism* of the job at any time but not its remaining work and parallelism, the algorithm is said to be *IP-clairvoyant*.

In any valid schedule, we require the total processor allocation at any time to be at most the total number of available processors, i.e., $\sum_{i=1}^{n} a_i(t) \leq P$. Let r_i denote the *release time* of job J_i. If all jobs are released together, their release time can be assumed to be all 0. Otherwise, we can assume without loss of generality that the first released job arrives at time 0. Let c_i denote the completion time of job J_i, and let c_i^k denote the completion time of phase J_i^k. We also require that a valid schedule cannot begin to execute a phase of a job unless it has completed all its preceding phases, i.e., $r_i = c_i^0 < c_i^1 < \cdots < c_i^{k_i} = c_i$, and $\int_{c_i^{k-1}}^{c_i^k} \Gamma_i^k(t)dt = w_i^k$ for all $1 \leq k \leq k_i$.

The *flow time* f_i of any job J_i is the duration between its completion and release, i.e., $f_i = c_i - r_i$. The *total flow time* $F(\mathcal{J})$ of all jobs in $\mathcal{J}$ is given by $F(\mathcal{J}) = \sum_{i=1}^{n} f_i$. The *makespan* $M(\mathcal{J})$ is the completion time of the last completed job, i.e., $M(\mathcal{J}) = \max_{i=1,\cdots,n} c_i$. Job J_i is said to be *active* at time t if it is released but not completed at t, i.e., $r_i \leq t \leq c_i$. An alternative expression for the total flow time is $F(\mathcal{J}) = \int_0^\infty n_t dt$, where n_t is the number of active jobs at time t. Let $u_i(t)$ denote the power consumed by job J_i at time t, i.e., $u_i(t) = \sum_{j=1}^{a_i(t)} s_{ij}(t)^\alpha$. The overall energy consumption e_i of the job is given by $e_i = \int_0^\infty u_i(t)dt$, and the total energy consumption $E(\mathcal{J})$ of the job set is $E(\mathcal{J}) = \sum_{i=1}^{n} e_i$, or alternatively $E(\mathcal{J}) = \int_0^\infty u_t dt$, where $u_t = \sum_{i=1}^{n} u_i(t)$ denotes the total power consumption of all jobs at time t. In this paper, we consider total flow time plus energy $G(\mathcal{J})$ and makespan plus energy $H(\mathcal{J})$ of the job set, i.e., $G(\mathcal{J}) = F(\mathcal{J}) + E(\mathcal{J})$ and $H(\mathcal{J}) = M(\mathcal{J}) + E(\mathcal{J})$. The objective is to minimize either $G(\mathcal{J})$ or $H(\mathcal{J})$.

We use *competitive analysis* [11] to evaluate an online scheduling algorithm by comparing its performance with that of an optimal offline scheduler. An online algorithm A is said to be c_1-*competitive* with respect to total flow time plus energy if it satisfies $G_{\mathsf{A}}(\mathcal{J}) \leq c_1 \cdot G_{\mathsf{OPT}}(\mathcal{J})$ for any job set $\mathcal{J}$, where $G_{\mathsf{OPT}}(\mathcal{J})$ denotes the total flow time plus energy of $\mathcal{J}$ under an optimal offline scheduler. Similarly, an online algorithm B is said to be c_2-*competitive* with respect to makespan plus energy if for any job set $\mathcal{J}$ we have $H_{\mathsf{B}}(\mathcal{J}) \leq c_2 \cdot H_{\mathsf{OPT}}(\mathcal{J})$, where $H_{\mathsf{OPT}}(\mathcal{J})$ denotes the makespan plus energy of the job set under an optimal offline scheduler.

3 Total Flow Time Plus Energy

We consider the objective of total flow time plus energy in this section. We first present a non-clairvoyant algorithm N-EQUI and analyze its performances for jobs with both arbitrary release time and the same release time. We then derive a lower bound on the competitive ratio of any non-clairvoyant algorithm. Finally, we present an IP-clairvoyant algorithm U-CEQ and show that it significantly improves upon any non-clairvoyant algorithm.

3.1 Preliminaries

We first derive two lower bounds on the total flow time plus energy of any job set, which allows us to bound the performance of our online algorithms through indirect comparisons instead of comparing directly to the optimal offline scheduler. We then introduce some useful notations, and outline the analysis techniques used to prove the competitiveness of our online algorithms.

3.1.1 Lower Bounds on Total Flow Time plus Energy

Without loss of generality, we assume that the jobs in a job set $\mathcal{J}$ are renamed in non-increasing order of total work, i.e., $w(J_1) \geq w(J_2) \geq \cdots \geq w(J_n)$. The following lemma gives two lower bounds $G_1^*(\mathcal{J})$ and $G_2^*(\mathcal{J})$ on the total flow time plus energy of job set $\mathcal{J}$. Note that the second lower bound only applies to a set of jobs with identical release time; an algorithm may incur a smaller total flow time plus energy than $G_2^*(\mathcal{J})$ if the jobs in $\mathcal{J}$ have arbitrary release time.

Lemma 1 *The total flow time plus energy of any job set $\mathcal{J}$ satisfies the following two lower bounds, i.e., $G_{\mathsf{OPT}}(\mathcal{J}) \geq \max\{G_1^*(\mathcal{J}), G_2^*(\mathcal{J})\}$,*

$$G_1^*(\mathcal{J}) = \frac{\alpha}{(\alpha-1)^{1-1/\alpha}} \sum_{i=1}^{n} x(J_i), \quad (1)$$

$$G_2^*(\mathcal{J}) = \frac{\alpha}{((\alpha-1)P)^{1-1/\alpha}} \sum_{i=1}^{n} i^{1-1/\alpha} \cdot w(J_i), \quad (2)$$

where $x(J_i)$ and $w(J_i)$ denote the unit-power span and the total work of job J_i, respectively, and P is the total number of processors. The second lower bound $G_2^*(\mathcal{J})$ only applies to jobs with identical release time.

Proof. To derive the first lower bound, consider any phase J_i^k of job J_i. The optimal scheduler will only perform better if there is an unlimited number of processors at its disposal. In this case, it will allocate a processors of the same speed, say s, to the phase throughout its execution, since the convexity of the power function implies that if different speeds are used, then averaging the speeds will result in the same execution rate but consuming less energy [49]. Moreover, we have $a \leq h_i^k$, since allocating more processors to a phase than its parallelism will incur more energy without improving flow time. The flow time plus energy introduced by the execution of J_i^k is then given by $\frac{w_i^k}{as} + \frac{w_i^k}{as} \cdot as^\alpha = w_i^k \left(\frac{1}{as} + s^{\alpha-1}\right) \geq \frac{\alpha}{(\alpha-1)^{1-1/\alpha}} \cdot \frac{w_i^k}{a^{1-1/\alpha}} \geq \frac{\alpha}{(\alpha-1)^{1-1/\alpha}} \cdot \frac{w_i^k}{(h_i^k)^{1-1/\alpha}} = \frac{\alpha}{(\alpha-1)^{1-1/\alpha}} \cdot x_i^k$. Extending this property over all phases and all jobs gives the first lower bound.

For the second lower bound, the optimal offline scheduler will do no worse in terms of total flow time plus energy if each job in the job set is replaced by a simpler job that contains a single fully-parallelizable phase with work w_i, since the original optimal schedule is also a valid schedule for the new job set. Also, because the jobs in the job set are assumed to have the same release time, it is well-known that the optimal offline scheduler will execute them using the SJF (Shortest Job First) policy, since otherwise the total flow time can be reduced by swapping the jobs without affecting the energy consumption. Moreover, for each job J_i, the optimal offline scheduler will allocate all P processors the same speed, say s, throughout its execution, by the same argument as in the proof of the first lower bound. The flow time plus energy introduced by the execution of J_i is then given by $\frac{w(J_i)}{Ps} \cdot i + \frac{w(J_i)}{Ps} \cdot Ps^\alpha = w(J_i)\left(\frac{i}{Ps} + s^{\alpha-1}\right) \geq \frac{\alpha}{(\alpha-1)^{1-1/\alpha}} \cdot \frac{i^{1-1/\alpha} \cdot w(J_i)}{P^{1-1/\alpha}}$. Summing the inequality over all jobs gives the second lower bound. $\square$

3.1.2 Concepts and Notations

We define some useful concepts and notations in this subsection in order to analyze the performance of any online algorithm A.

First, let $\mathcal{J}(t)$ and $\mathcal{J}^*(t)$ denote the sets of active jobs at any time t scheduled by an online algorithm A and the optimal offline algorithm, respectively. Since an online algorithm and the optimal algorithm may schedule the same job set differently, $\mathcal{J}(t)$ and $\mathcal{J}^*(t)$ can be different from each other at any time instance. We define $\frac{dG_{\mathsf{A}}(\mathcal{J}(t))}{dt}$ to be the *instantaneous cost* of the online algorithm A with respect to the total flow time plus energy at time t, and define $\frac{dG_{\mathsf{OPT}}(\mathcal{J}^*(t))}{dt}$ to be the instantaneous cost of the optimal offline algorithm at time t. Since the online algorithm contains n_t active jobs and consumes u_t power at time t, its instantaneous cost is given by $\frac{dG_{\mathsf{A}}(\mathcal{J}(t))}{dt} = n_t + u_t$, and its total flow time plus energy for the entire job set can be obtained by integrating the above instantaneous cost over time, i.e., $G_A(\mathcal{J}) = \int_0^\infty \frac{dG_{\mathsf{A}}(\mathcal{J}(t))}{dt} dt = \int_0^\infty (n_t + u_t) dt$. Similarly, we have $\frac{dG_{\mathsf{OPT}}(\mathcal{J}^*(t))}{dt} = n_t^* + u_t^*$ for the optimal offline algorithm, where n_t^* and u_t^* denote the number of active jobs and the power consumption under the optimal algorithm at time t. The total flow time plus energy incurred by the optimal offline algorithm is then given by $G_{\mathsf{OPT}}(\mathcal{J}) = \int_0^\infty \frac{dG_{\mathsf{OPT}}(\mathcal{J}^*(t))}{dt} dt = \int_0^\infty (n_t^* + u_t^*) dt$.

Now, let us define the notions of t-prefix and t-suffix. Specifically, the *t-prefix* $J_i(\overleftarrow{t})$ of a job J_i is defined as the portion of the job scheduled by the online algorithm A no later than time t, and the *t-suffix* $J_i(\overrightarrow{t})$ is the portion of job J_i scheduled by algorithm A after time t. Moreover, we can extend the notions of t-prefix and t-suffix from an individual job to a job set as follows. The t-prefix of a job set $\mathcal{J}$ scheduled by the online algorithm A is defined as $\mathcal{J}(\overleftarrow{t}) = \{J_i(\overleftarrow{t}) : J_i \in \mathcal{J} \text{ and } r_i \leq t\}$ and the t-suffix of job set $\mathcal{J}$ is defined as $\mathcal{J}(\overrightarrow{t}) = \{J_i(\overrightarrow{t}) : J_i \in \mathcal{J} \text{ and } r_i \leq t\}$. With the help of these notions, we define $\frac{dG_1^*(\mathcal{J}(t))}{dt} = \frac{G_1^*(\mathcal{J}(\overleftarrow{t+\Delta t})) - G_1^*(\mathcal{J}(\overleftarrow{t}))}{\Delta t}$ and $\frac{dG_2^*(\mathcal{J}(t))}{dt} = \frac{G_2^*(\mathcal{J}(\overleftarrow{t+\Delta t})) - G_2^*(\mathcal{J}(\overleftarrow{t}))}{\Delta t}$ to be the rates of change at any time t for the two lower bounds presented in Lemma 1, where Δt represents an infinitesimally small interval of time during which no job arrives or completes. Note that these two rates of change are defined with respect to the t-prefix, or the completed portion, of the job set $\mathcal{J}$ scheduled by the online algorithm A. From the definitions of the two lower bounds, we can see that $\frac{dG_1^*(\mathcal{J}(t))}{dt}$ and $\frac{dG_2^*(\mathcal{J}(t))}{dt}$ are essentially determined by how much work is done and how much unit-power span is completed for the jobs at time t under the online algorithm A. For instance, if algorithm A does not schedule any job at time t, the t-prefix of the job set will not change, i.e., $\mathcal{J}(\overleftarrow{t+\Delta t}) = \mathcal{J}(\overleftarrow{t})$, and as a result we will have

$\frac{dG_1^*(\mathcal{J}(t))}{dt} = \frac{dG_2^*(\mathcal{J}(t))}{dt} = 0$. If an online algorithm always schedules some job at all times, the following lemma relates the two rates of change to the two lower bounds.

Lemma 2 *For any job set $\mathcal{J}$ scheduled by an online algorithm A, which always schedules some job at all times, it satisfies that $G_1^*(\mathcal{J}) = \int_0^\infty \frac{dG_1^*(\mathcal{J}(t))}{dt} dt$ and $G_2^*(\mathcal{J}) = \int_0^\infty \frac{dG_2^*(\mathcal{J}(t))}{dt} dt$.*

Proof. According to definition, the t-prefix $J_i(\overleftarrow{t})$ of a job J_i does not belong to $\mathcal{J}(\overleftarrow{t})$ if the job is not yet released at time t, but the job will remain in $\mathcal{J}(\overleftarrow{t})$ even after it has been completed. This ensures that the completion of a job will not decrease $\frac{dG_1^*(\mathcal{J}(t))}{dt}$ and $\frac{dG_2^*(\mathcal{J}(t))}{dt}$, so they are non-negative at all times. Since the online algorithm A always schedules some active job, all jobs are guaranteed to complete in finite amount of time. Thus, we can express the two lower bounds by integrating their rates of change over time, i.e., $G_1^*(\mathcal{J}) = \int_0^\infty \frac{dG_1^*(\mathcal{J}(t))}{dt} dt$ and $G_2^*(\mathcal{J}) = \int_0^\infty \frac{dG_2^*(\mathcal{J}(t))}{dt} dt$. □

Finally, for analyzing the performance of an online algorithm A for a set of jobs with arbitrary release time, we also need to define a potential function $\Phi(t)$, whose form is usually associated with the status of the job set at any time t under both online algorithm and the optimal offline algorithm [30]. We can similarly define $\frac{d\Phi(t)}{dt} = \frac{\Phi(t+\Delta t)-\Phi(t)}{\Delta t}$ to be the rate of change for the potential function at time t.

3.1.3 Analysis Techniques

We now outline two analysis techniques for proving the competitiveness of any online scheduling algorithm. They are commonly known as the amortized local competitiveness argument and the local competitiveness argument in the literature [39, 30]. Both techniques compare the cost of an online algorithm at any local time instance, or its instantaneous cost, with respect to that of an optimal offline scheduler. For arbitrarily released jobs, the comparison is performed with the help of the first lower bound given in Lemma 1 and a carefully designed potential function. For jobs with identical release time, both lower bounds are used to represent the performance of the optimal.

The following lemma first illustrates the use of *amortized local competitiveness argument* for jobs with arbitrary release time. The technique arrives at the competitive ratio of any online algorithm A by bounding its instantaneous cost at any time t with respect to the optimal offline scheduler.

Lemma 3 *Suppose that an online algorithm A schedules a set $\mathcal{J}$ of jobs with arbitrary release time. Then A is $(c_1 + c_2)$-competitive with respect to total flow time plus energy, if given a potential function $\Phi(t)$, the execution of the job set satisfies the following*

- Boundary condition: $\Phi(0) \le 0$ and $\Phi(\infty) \ge 0$;

- Arrival condition: $\Phi(t)$ does not increase whenever a new job arrives;

- Completion condition: $\Phi(t)$ does not increase whenever a job completes under either A or the optimal offline scheduler;

- Running condition: $\frac{dG_{\mathsf{A}}(\mathcal{J}(t))}{dt} + \frac{d\Phi(t)}{dt} \le c_1 \cdot \frac{dG_{\mathsf{OPT}}(\mathcal{J}^*(t))}{dt} + c_2 \cdot \frac{dG_1^*(\mathcal{J}(t))}{dt}$ at all time $t \ge 0$.

Proof. Let T denote the set of time instances when a job arrives or completes under either the online algorithm A or the optimal offline scheduler. Integrating the running condition over time and applying Lemma 2, we get $G_{\mathsf{A}}(\mathcal{J}) + \Phi(\infty) - \Phi(0) + \sum_{t\in T} (\Phi(t^-) - \Phi(t^+)) \le c_1 \cdot G_{\mathsf{OPT}}(\mathcal{J}) + c_2 \cdot G_1^*(\mathcal{J})$, where t^- and t^+ denote the times right before and after the event occurred at time t. Now, applying boundary, arrival and completion conditions to the above inequality, we get $G_{\mathsf{A}}(\mathcal{J}) \leq c_1 \cdot G_{\mathsf{OPT}}(\mathcal{J}) + c_2 \cdot G_1^*(\mathcal{J})$. Since $G_1^*(\mathcal{J})$ is a lower bound on the total flow time plus energy of job set $\mathcal{J}$ according to Lemma 1, the performance of algorithm A satisfies $G_{\mathsf{A}}(\mathcal{J}) \leq (c_1 + c_2) \cdot G_{\mathsf{OPT}}(\mathcal{J})$. □

When scheduling for a set of jobs with identical release time, the analysis turns out to be simpler as the potential function is usually not needed. In this case, we can get the competitive ratio of online algorithm A by using the *local competitiveness argument*, which directly compares its instantaneous cost at any time t with respect to the rates of change for both lower bounds given in Lemma 1. The following lemma illustrates this technique.

Lemma 4 *Suppose that an online algorithm* A *schedules a set* $\mathcal{J}$ *of jobs with identical release time. Then* A *is* $(c_1 + c_2)$*-competitive with respect to total flow time plus energy, if the execution of the job set satisfies the following*
- Running condition: $\frac{dG_{\mathsf{A}}(\mathcal{J}(t))}{dt} \leq c_1 \cdot \frac{dG_1^*(\mathcal{J}(t))}{dt} + c_2 \cdot \frac{dG_2^*(\mathcal{J}(t))}{dt}$ *at all time* $t \geq 0$.

Proof. Similarly to the proof of Lemma 3, by integrating the running condition over time and applying Lemma 2, we get $G_{\mathsf{A}}(\mathcal{J}) \leq c_1 \cdot G_1^*(\mathcal{J}) + c_2 \cdot G_2^*(\mathcal{J})$. Since both $G_1^*(\mathcal{J})$ and $G_2^*(\mathcal{J})$ are lower bounds on the total flow time plus energy of job set $\mathcal{J}$ according to Lemma 1, the result follows. □

3.2 Non-clairvoyant Algorithm: N-EQUI

It was shown in [17, 44] that any non-clairvoyant algorithm which allocates a set of uniform-speed processors to a job is $\Omega(P^{(\alpha-1)/\alpha^2})$-competitive, where P is the total number of processors. To achieve better performance, we propose a non-clairvoyant algorithm called N-EQUI (Non-uniform Equi-partitioning), which equally partitions the P processors among the n_t active jobs at any time t. Algorithm 1 describes its details.

Specifically, when the number of processors is at least the number of active jobs, i.e., $P \geq n_t$, it sets the speeds of the allocated processors for each active job in a non-uniform manner. Intuitively, since the algorithm does not know the parallelism of a job to guide its processor allocation, the non-uniform speed assignment balances the waste of energy due to the possible overallocation and the delay of the job due to the possible underallocation. On the other hand, when $n_t > P$, it assigns the same speed to all processors and relies on time-sharing to allocate P/n_t fraction of a processor to each active job, which is commonly implemented in the operating system using the round robin policy. Note that since the parallelism of any active job is at least 1, no energy waste will be incurred in this case.

At any time t when job J_i is in its k'th phase, we say that it is *satisfied* if its processor allocation is at least its instantaneous parallelism, i.e., $a_i(t) \geq h_i^k$. Otherwise, the job is said to be *deprived*. Let $\mathcal{J}(t)$ denote the set of all active jobs at time t, and let $\mathcal{J}_S(t)$ and $\mathcal{J}_D(t)$ denote the set of satisfied and deprived jobs at time t, respectively. For convenience, we let $n_t^S = |\mathcal{J}_S(t)|$ and $n_t^D = |\mathcal{J}_D(t)|$. Since an active job is either satisfied or deprived, we have $|\mathcal{J}(t)| = n_t = n_t^S + n_t^D$. Moreover, we define $x_t = n_t^D / n_t$ to be the *deprived ratio* at time t. To assist analysis, we first bound the execution rate and the power consumption of N-EQUI for any active job at time t in the following lemma.

Lemma 5 *Suppose that* N-EQUI *schedules a set* $\mathcal{J}$ *of jobs. Then for any job* $J_i \in \mathcal{J}$*, its execution rate at time t satisfies* $\left(\frac{1}{(\alpha-1)H_P}\right)^{1/\alpha} \frac{\bar{a}_i(t)^{1-1/\alpha}}{2^{1/\alpha}} \leq \Gamma_i^k(t) \leq \left(\frac{1}{(\alpha-1)H_P}\right)^{1/\alpha} \frac{\bar{a}_i(t)^{1-1/\alpha}}{1-1/\alpha}$*, where* $\bar{a}_i(t) = \min\{a_i(t), h_i^k\}$ *denotes the effective processor allocation for job* J_i *at time t. Also, the power consumption of the job at time t satisfies* $u_i(t) \leq \frac{1}{\alpha-1}$.

Algorithm 1 N-EQUI

```
Input: total number $P$ of processors and number $n_t$ of active jobs at time $t$.
Output: number of allocated processors and their speeds for each active job at time $t$.
1: if $P \geq n_t$ then
2:     allocate $a_i(t) = \left\lfloor \frac{P}{n_t} \right\rfloor$ processors to each active job $J_i$.
3:     set the speed of the $j$'th processor allocated to job $J_i$ to be $s_{ij}(t) = \left(\frac{1}{(\alpha-1)H_P \cdot j}\right)^{1/\alpha}$,
       where $j = 1, \cdots, a_i(t)$ and $H_P = \sum_{k=1}^{P} \frac{1}{k}$ is the $P$'th harmonic number.
4: else
5:     allocate $a_i(t) = \frac{P}{n_t}$ fraction of a processor to each active job $J_i$.
6:     set the speed of all processors to be $s(t) = \left(\frac{n_t}{(\alpha-1)H_P \cdot P}\right)^{1/\alpha}$.
7: end if
```

Proof. When $P < n_t$, we have $a_i(t) = P/n_t < 1 \leq h_i^k$ for job J_i, so $\bar{a}_i(t) = a_i(t)$. The execution rate of the job is given by $\Gamma_i^k(t) = a_i(t)s(t) = \left(\frac{1}{(\alpha-1)H_P}\right)^{1/\alpha} \bar{a}_i(t)^{1-1/\alpha}$, and its power consumed is $u_i(t) = \frac{1}{(\alpha-1)H_P} \leq \frac{1}{\alpha-1}$.

When $P \geq n_t$, we have $\bar{a}_i(t) \geq 1$, since $a_i(t) = \lfloor P/n_t \rfloor \geq 1$ and $h_i^k \geq 1$. The execution rate of the job is $\Gamma_i^k(t) = \sum_{j=1}^{\bar{a}_i(t)} s_{ij}(t) = \left(\frac{1}{(\alpha-1)H_P}\right)^{1/\alpha} \sum_{j=1}^{\bar{a}_i(t)} \frac{1}{j^{1/\alpha}}$, which can be approximated with integration: $\sum_{j=1}^{\bar{a}_i(t)} s_{ij}(t) \leq \left(\frac{1}{(\alpha-1)H_P}\right)^{1/\alpha} \int_0^{\bar{a}_i(t)} \frac{1}{j^{1/\alpha}} dj = \left(\frac{1}{(\alpha-1)H_P}\right)^{1/\alpha} \frac{\bar{a}_i(t)^{1-1/\alpha}}{1-1/\alpha}$, and $\sum_{j=1}^{\bar{a}_i(t)} s_{ij}(t) \geq \left(\frac{1}{(\alpha-1)H_P}\right)^{1/\alpha} \int_1^{\bar{a}_i(t)+1} \frac{1}{j^{1/\alpha}} dj = \left(\frac{1}{(\alpha-1)H_P}\right)^{1/\alpha} \frac{(\bar{a}_i(t)+1)^{1-1/\alpha}-1}{1-1/\alpha} \geq \left(\frac{1}{(\alpha-1)H_P}\right)^{1/\alpha} \cdot \frac{2^{1-1/\alpha}-1}{1-1/\alpha} \bar{a}_i(t)^{1-1/\alpha} \geq \left(\frac{1}{(\alpha-1)H_P}\right)^{1/\alpha} \frac{\bar{a}_i(t)^{1-1/\alpha}}{2^{1/\alpha}}$. The second to last inequality follows because $\frac{(x+1)^{1-1/\alpha}-1}{x^{1-1/\alpha}}$ is an increasing function of x for all $x > 0$ and we have $\bar{a}_i(t) \geq 1$. The power consumption of the job satisfies $u_i(t) = \sum_{j=1}^{a_i(t)} s_{ij}(t)^\alpha = \frac{1}{(\alpha-1)H_P} \sum_{j=1}^{a_i(t)} \frac{1}{j} = \frac{H_{a_i(t)}}{(\alpha-1)H_P} \leq \frac{1}{\alpha-1}$, where $H_{a_i(t)}$ is $a_i(t)$'th harmonic number, and $H_{a_i(t)} \leq H_P$ since $a_i(t) \leq P$ at all times. $\square$

3.2.1 Performance for Jobs with Arbitrary Release Time

We first bound the performance of N-EQUI for a set of jobs with arbitrary release time. We adopt the potential function proposed by Lam et al. [37] in the analysis of an online speed scaling algorithm for sequential jobs. Specifically, we focus on the t-suffix $\mathcal{J}(\overrightarrow{t})$ of job set $\mathcal{J}$ and define $n_t(z)$ to be the number of active jobs whose remaining work is at least z at time t under N-EQUI, i.e., $n_t(z) = \sum_{J_i \in \mathcal{J}(t)} [w(J_i(\overrightarrow{t})) \geq z]$, where $[x]$ is 1 if proposition x is true and 0 otherwise. Also, define $n_t^*(z)$ to be the number of active jobs whose remaining work is at least z at time t under the optimal offline algorithm. The potential function is then defined as

$$\Phi(t) = \eta \int_0^\infty \left[\left(\sum_{i=1}^{n_t(z)} i^{1-1/\alpha} \right) - n_t(z)^{1-1/\alpha} n_t^*(z) \right] dz, \tag{3}$$

where $\eta = \eta' \frac{H_P^{1/\alpha}}{P^{1-1/\alpha}}$ and η' is a constant to be specified later. In particular, the integration of the first term of the potential function is proportional to the optimal total flow time plus energy for the remaining portion, or t-suffix, of the job set at time t scheduled under N-EQUI. The second term of the potential function is added to ensure that the arrival condition will be satisfied. (See proof of Theorem 1.)

In addition, we need to use the following lemma in our proof.

Lemma 6 *For any $n_t \geq 0$, $s_j^* \geq 0$ and $\lambda > 0$, we have that $n_t^{1-1/\alpha} s_j^* \leq \frac{\lambda(H_P \cdot P)^{1-1/\alpha}}{\alpha}\left(s_j^*\right)^\alpha + \frac{1-1/\alpha}{\lambda^{1/(\alpha-1)}(H_P \cdot P)^{1/\alpha}} n_t$.*

Proof. The lemma is a direct result of Young's Inequality [27], which is stated formally as follows. If f is a continuous and strictly increasing function on $[0, c]$ with $c > 0$, $f(0) = 0$, $a \in [0, c]$ and $b \in [0, f(c)]$, then $ab \leq \int_0^a f(x)dx + \int_0^b f^{-1}(x)dx$, where f^{-1} is the inverse function of f. By setting $f(x) = \lambda\left(H_P \cdot P\right)^{1-1/\alpha} x^{\alpha-1}$, $a = s_j^*$ and $b = n_t^{1-1/\alpha}$, the lemma is directly implied. □

Using Lemmas 3, 5 and 6, we now prove the competitive ratio of N-EQUI for jobs with arbitrary release time.

Theorem 1 N-EQUI *is $O(\ln P)$-competitive with respect to total flow time plus energy for any set of parallel jobs, where P is the total number of processors.*

Proof. We will show that the execution of any job set scheduled by N-EQUI (NE for short) satisfies the boundary, arrival and completion conditions in Lemma 3, as well as the running condition $\frac{dG_{\mathsf{NE}}(\mathcal{J}(t))}{dt} + \frac{d\Phi(t)}{dt} \leq c_1 \cdot \frac{dG_{\mathsf{OPT}}(\mathcal{J}^*(t))}{dt} + c_2 \cdot \frac{dG_1^*(\mathcal{J}(t))}{dt}$, where $c_1 = O(\ln P)$ and $c_2 = O(\ln^{1/\alpha} P)$. The theorem then follows by Lemma 3.

- *Boundary condition*: At time 0, no job exists, so we have $n_t(z) = n_t^*(z) = 0$ for all $z \geq 0$, and so $\Phi(0) = 0$. At time ∞, all jobs are completed, so again we have $\Phi(\infty) = 0$. Hence, the boundary condition is satisfied.

- *Arrival condition*: Let t^- and t^+ denote the time instances right before and after a new job with work w arrives at time t. Then we have $n_{t^+}(z) = n_{t^-}(z) + 1$ for $z \leq w$ and $n_{t^+}(z) = n_{t^-}(z)$ for $z > w$, and similarly $n_{t^+}^*(z) = n_{t^-}^*(z) + 1$ for $z \leq w$ and $n_{t^+}^*(z) = n_{t^-}^*(z)$ for $z > w$. For convenience, we define $\phi_t(z) = \left(\sum_{i=1}^{n_t(z)} i^{1-1/\alpha}\right) - n_t(z)^{1-1/\alpha} n_t^*(z)$. It is obvious that for $z > w$, we have $\phi_{t^+}(z) = \phi_{t^-}(z)$. For $z \leq w$, we can get $\phi_{t^+}(z) - \phi_{t^-}(z) = n_{t^-}^*(z)\left(n_{t^-}(z)^{1-1/\alpha} - (n_{t^-}(z)+1)^{1-1/\alpha}\right) \leq 0$. Hence, $\Phi(t^+) = \eta\int_0^\infty \phi_{t^+}(z)dz \leq \eta\int_0^\infty \phi_{t^-}(z)dz = \Phi(t^-)$, and the arrival condition is satisfied.

- *Completion condition*: When a job completes under either N-EQUI or the optimal schedule, $\Phi(t)$ is unchanged because $n_t(z)$ or $n_t^*(z)$ reduces by 1 for $z = 0$ but does not change for all $z > 0$. Therefore, the completion condition is satisfied.

- *Running condition*: According to Lemma 5, the overall power consumption u_t of all active jobs at time t under N-EQUI satisfies $u_t \leq \frac{n_t}{\alpha-1}$. Thus, we have $\frac{dG_{\mathsf{NE}}(\mathcal{J}(t))}{dt} = n_t + u_t \leq \frac{\alpha}{\alpha-1} n_t$. Suppose the optimal offline scheduler sets the speed of the j'th processor to s_j^* at time t, which then gives $\frac{dG_{\mathsf{OPT}}(\mathcal{J}^*(t))}{dt} = n_t^* + u_t^* = n_t^* + \sum_{j=1}^P \left(s_j^*\right)^\alpha$. To bound the rate of change $\frac{dG_1^*(\mathcal{J}(t))}{dt}$, it turns out to be sufficient to consider the set $J_S(t)$ of satisfied jobs. Specifically, for each satisfied job $J_i \in J_S(t)$, if it is in its k'th phase at time t under N-EQUI, then we have $a_i(t) \geq h_i^k$. According to Lemma 5, the execution rate of the job is given by $\Gamma_i^k(t) \geq \left(\frac{1}{(\alpha-1)H_P}\right)^{1/\alpha} \frac{\left(h_i^k\right)^{1-1/\alpha}}{2^{1/\alpha}}$. Since $\frac{dG_1^*(\mathcal{J}(t))}{dt}$ only depends on the unit-power span completed for the jobs at time t under the schedule of N-EQUI, we have

$\frac{dG_1^*(\mathcal{J}(t))}{dt} \geq \frac{\alpha}{(\alpha-1)^{1-1/\alpha}} \sum_{J_i \in \mathcal{J}_S(t)} \frac{\Gamma_i^k(t)}{\left(h_i^k\right)^{1-1/\alpha}} \geq \frac{\alpha}{\alpha-1}\left(\frac{1}{2H_P}\right)^{1/\alpha} n_t^S = \frac{\alpha}{\alpha-1}\left(\frac{1}{2H_P}\right)^{1/\alpha}(1-x_t)n_t$, where x_t is the deprived ratio.

Now, we focus on finding an upper bound for the rate of change $\frac{d\Phi(t)}{dt}$ of the potential function $\Phi(t)$ at time t. To this end, we mainly consider the set $\mathcal{J}_D(t)$ of deprived jobs. If a deprived job $J_i \in \mathcal{J}_D(t)$ is in its k'th phase at time t under N-EQUI, then $a_i(t) < h_i^k$, and the execution rate of the job satisfies $\left(\frac{1}{(\alpha-1)H_P}\right)^{1/\alpha} \frac{a_i(t)^{1-1/\alpha}}{2^{1/\alpha}} \leq \Gamma_i^k(t) \leq \left(\frac{1}{(\alpha-1)H_P}\right)^{1/\alpha} \frac{a_i(t)^{1-1/\alpha}}{1-1/\alpha}$ by Lemma 5. In the worst case, the n_t^D deprived jobs may have the most remaining work, so they will take the smallest n_t^D indices in the first term of the potential function given in Equation (3). While considering the set of satisfied jobs can further decrease $\Phi(t)$, we ignore these jobs for deriving an upper bound on the rate of change for the first term of the potential function, and use all active jobs for bounding the rate of change for the second term. The overall rate of change for the potential function can then bounded by

$$\begin{aligned}
\frac{d\Phi(t)}{dt} &= \eta \cdot \frac{d}{dt}\int_0^\infty \left[\left(\sum_{i=1}^{n_t(z)} i^{1-1/\alpha}\right) - n_t(z)^{1-1/\alpha} n_t^*(z)\right] dz \\
&\leq \frac{\eta}{\Delta t}\int_0^\infty \left[\left(\sum_{i=1}^{n_{t+\Delta t}(z)} i^{1-1/\alpha}\right) - \left(\sum_{i=1}^{n_t(z)} i^{1-1/\alpha}\right)\right] dz \\
&\quad + \frac{\eta}{\Delta t}\int_0^\infty \left[n_t(z)^{1-1/\alpha}\left(n_t^*(z) - n_{t+\Delta t}^*(z)\right) + n_t^*(z)\left(n_t(z)^{1-1/\alpha} - n_{t+\Delta t}(z)^{1-1/\alpha}\right)\right] dz \\
&\leq \frac{\eta' H_P^{1/\alpha}}{P^{1-1/\alpha}}\left(-\sum_{i=1}^{n_t^D} i^{1-1/\alpha}\cdot\Gamma_i^k(t) + n_t^{1-1/\alpha}\sum_{j=1}^{P} s_j^* + n_t^*\sum_{i=1}^{n_t}\left(i^{1-1/\alpha} - (i-1)^{1-1/\alpha}\right)\Gamma_i^k(t)\right).
\end{aligned}$$

We can get $\sum_{i=1}^{n_t^D} i^{1-1/\alpha} \geq \int_0^{n_t^D} i^{1-1/\alpha} di = \frac{\left(n_t^D\right)^{2-1/\alpha}}{2-1/\alpha} \geq \frac{x_t^2 n_t^{2-1/\alpha}}{2}$ and $\sum_{i=1}^{n_t}\left(i^{1-1/\alpha} - (i-1)^{1-1/\alpha}\right) = n_t^{1-1/\alpha}$. Moreover, according to Lemma 6, we have $n_t^{1-1/\alpha}\sum_{j=1}^P s_j^* \leq \frac{\lambda(H_P\cdot P)^{1-1/\alpha}}{\alpha}\sum_{j=1}^P\left(s_j^*\right)^\alpha + \frac{1-1/\alpha}{\lambda^{1/(\alpha-1)}(H_P\cdot P)^{1/\alpha}}Pn_t$, where λ is a constant to be specified later. For each satisfied job $J_i \in J_S(t)$, we have $h_i^k \leq a_i(t)$, so its execution rate satisfies $\Gamma_i^k(t) \leq \left(\frac{1}{(\alpha-1)H_P}\right)^{1/\alpha}\frac{\left(h_i^k\right)^{1-1/\alpha}}{1-1/\alpha} \leq \left(\frac{1}{(\alpha-1)H_P}\right)^{1/\alpha}\frac{a_i(t)^{1-1/\alpha}}{1-1/\alpha}$ by Lemma 5. Finally, for each job $J_i \in \mathcal{J}(t)$, we have $\frac{P}{2n_t} \leq \left\lfloor \frac{P}{n_t} \right\rfloor \leq a_i(t) \leq \frac{P}{n_t}$. Substituting these bounds into $\frac{d\Phi(t)}{dt}$ above and simplifying, we have

$$\frac{d\Phi(t)}{dt} \leq \eta'\left(-\frac{x_t^2}{4(\alpha-1)^{1/\alpha}}n_t + \frac{\lambda H_P}{\alpha}\sum_{j=1}^P\left(s_j^*\right)^\alpha + \frac{1-1/\alpha}{\lambda^{1/(\alpha-1)}}n_t + \frac{\alpha}{(\alpha-1)^{1+1/\alpha}}n_t^*\right). \quad (4)$$

Now, we set $\eta' = \frac{4\alpha^2}{(\alpha-1)^{1-1/\alpha}}$ and $\lambda = 4^{\alpha-1}(\alpha-1)^{1-1/\alpha}$. Substituting Inequality (4) as well as the bounds for $\frac{dG_{\mathsf{NE}}(\mathcal{J}(t))}{dt}$, $\frac{dG_{\mathsf{OPT}}(\mathcal{J}^*(t))}{dt}$ and $\frac{dG_1^*(\mathcal{J}(t))}{dt}$ into the running condition, we can see that it can be satisfied for all valid values of x_t by setting the multipliers to be $c_1 = \max\{\frac{4\alpha^3}{(\alpha-1)^2}, 4^\alpha \alpha H_P\}$ and $c_2 = 2\alpha\cdot(2H_P)^{1/\alpha}$. Since α can be considered as a constant with respect to P, and it is well-known that $H_P = O(\ln P)$, the theorem is proved. $\square$

3.2.2 Performance for Jobs with Identical Release Time

We now bound the performance of N-EQUI for jobs with identical release time. We show that the competitive ratio of N-EQUI can be slightly improved in this case compared to the one achieved for arbitrarily released jobs. The following theorem gives the result.

Theorem 2 N-EQUI *is $O(\ln^{1/\alpha} P)$-competitive with respect to total flow time plus energy for any set of parallel jobs with identical release time, where P is the total number of processors.*

Proof. We will show that the execution of any job set scheduled by N-EQUI satisfies the running condition $\frac{dG_{\mathsf{NE}}(\mathcal{J}(t))}{dt} \le c_1 \cdot \frac{dG_1^*(\mathcal{J}(t))}{dt} + c_2 \cdot \frac{dG_2^*(\mathcal{J}(t))}{dt}$, where $c_1 = O(\ln^{1/\alpha} P)$ and $c_2 = O(\ln^{1/\alpha} P)$. The theorem is then implied by Lemma 4.

We first note from the proof of Theorem 1 that the instantaneous cost of N-EQUI satisfies $\frac{dG_{\mathsf{NE}}(\mathcal{J}(t))}{dt} \le \frac{\alpha}{\alpha-1} n_t$, and the rate of change for the first lower bound satisfies $\frac{dG_1^*(\mathcal{J}(t))}{dt} \ge \frac{\alpha}{\alpha-1}\left(\frac{1}{2H_P}\right)^{1/\alpha}(1-x_t)n_t$. It remains to bound the rate of change $\frac{dG_2^*(\mathcal{J}(t))}{dt}$ for the second lower bound. To this end, we focus on the t-prefix $\mathcal{J}(\overleftarrow{t})$ of the job set $\mathcal{J}$ and redefine $n_t(z)$ to be the number of jobs whose completed work is at least z at time t under N-EQUI, i.e., $n_t(z) = \sum_{J_i \in \mathcal{J}} [w(J_i(\overleftarrow{t})) \ge z]$. Note that we do not restrict a job to be active when considering its contribution towards $n_t(z)$. This is consistent with the definition of t-prefix $\mathcal{J}(\overleftarrow{t})$ for job set $\mathcal{J}$, which allows us to express the the second lower bound given in Lemma 1 for the t-prefix $\mathcal{J}(\overleftarrow{t})$ of job set $\mathcal{J}$ as follows

$$G_2^*(\mathcal{J}(\overleftarrow{t})) = \frac{\alpha}{((\alpha-1)P)^{1-1/\alpha}} \int_0^\infty \left(\sum_{i=1}^{n_t(z)} i^{1-1/\alpha}\right) dz.$$

Again we only focus on the set $\mathcal{J}_D(t)$ of deprived jobs, and for each deprived job $J_i \in \mathcal{J}_D(t)$, its execution rate satisfies $\Gamma_i^k(t) \ge \left(\frac{1}{(\alpha-1)H_P}\right)^{1/\alpha} \frac{a_i(t)^{1-1/\alpha}}{2^{1/\alpha}}$ by Lemma 5. In the worst case, the n_t^D deprived jobs have completed the most work so far, so they will take the smallest n_t^D indices in the above expression of $G_2^*(\mathcal{J}(\overleftarrow{t}))$. Thus, the rate of change $\frac{dG_2^*(\mathcal{J}(t))}{dt}$ is bounded by

$$\begin{aligned}
\frac{dG_2^*(\mathcal{J}(t))}{dt} &= \frac{\alpha}{((\alpha-1)P)^{1-1/\alpha}} \cdot \frac{d}{dt}\int_0^\infty \left(\sum_{i=1}^{n_t(z)} i^{1-1/\alpha}\right) dz \\
&= \frac{\alpha}{((\alpha-1)P)^{1-1/\alpha}} \cdot \frac{1}{\Delta t}\int_0^\infty \left[\left(\sum_{i=1}^{n_{t+\Delta t}(z)} i^{1-1/\alpha}\right) - \left(\sum_{i=1}^{n_t(z)} i^{1-1/\alpha}\right)\right] dz \\
&\ge \frac{\alpha}{((\alpha-1)P)^{1-1/\alpha}} \sum_{i=1}^{n_t^D} i^{1-1/\alpha} \cdot \Gamma_i^k(t) \\
&\ge \frac{\alpha}{\alpha-1} \cdot \frac{x_t^2 n_t}{4H_P^{1/\alpha}}. \qquad (5)
\end{aligned}$$

Substituting Inequality (5) as well as the bounds for $\frac{dG_{\mathsf{NE}}(\mathcal{J}(t))}{dt}$ and $\frac{dG_1^*(\mathcal{J}(t))}{dt}$ into the running condition, we can see that it is satisfied for all valid values of x_t if we set $c_1 = 2^{1+1/\alpha}H_P^{1/\alpha}$ and $c_2 = 4H_P^{1/\alpha}$. Thus, the theorem is proved. □

3.3 Lower Bound for Any Non-clairvoyant Algorithm

In this section, we prove a lower bound of $\Omega(\ln^{1/\alpha} P)$ on the competitive ratio of any non-clairvoyant algorithm even with non-uniform speed assignments. Since this lower bound matches the upper bound of N-EQUI for parallel jobs with identical release time, it shows that N-EQUI is asymptotically optimal in that setting.

Before proving the lower bound, we first present a useful lemma, which gives the solution of a minimization problem. The proof basically transforms this minimization problem into a convex optimization problem, and solves it by applying the KKT conditions [12].

Lemma 7 *For any $P \geq 1$, $\alpha > 1$ and $b > 0$, if $\sum_j^P s_j^\alpha = b$ and $s_1 \geq s_2 \geq \cdots \geq s_P \geq 0$, then $\max_{1 \leq h \leq P} \frac{h^{1-1/\alpha}}{\sum_{j=1}^h s_j}$ is minimized when $(s_1, s_2, \cdots, s_P)$ satisfy $\frac{h^{1-1/\alpha}}{\sum_{j=1}^h s_j} = \frac{(h-1)^{1-1/\alpha}}{\sum_{j=1}^{h-1} s_j}$ for all $h = 2, \cdots, P$.*

Proof. To prove this lemma, we transform the stated problem into a convex optimization problem. We then show that our proposed solution satisfies the KKT conditions [12], which are known to be sufficient for the optimality of convex minimization problems. This then leads to the proof of the lemma. First, by introducing a variable y, the original optimization problem can be transformed into the following minimization problem:

$$\begin{aligned} \text{minimize} \quad & y \\ \text{subject to} \quad & \sum_{j=1}^{P} s_j^\alpha = b \qquad (6) \\ & s_j \geq s_{j+1} \text{ for } j = 1, \cdots, P-1 \\ & y \geq \frac{h^{1-1/\alpha}}{\sum_{j=1}^h s_j} \text{ for } h = 1, \cdots, P \end{aligned}$$

However, the above minimization problem is not convex because its equality constraint (Equation 6) is not linear. Substituting $z_j = s_j^\alpha$, we transform it into a convex optimization problem as follows.

$$\begin{aligned} \text{minimize} \quad & y \\ \text{subject to} \quad & \sum_{j=1}^{P} z_j = b \qquad (7) \\ & z_{j+1} - z_j \leq 0 \text{ for } j = 1, \cdots, P-1 \qquad (8) \\ & \frac{h^{1-1/\alpha}}{\sum_{j=1}^h z_j^{1/\alpha}} - y \leq 0 \text{ for } h = 1, \cdots, P \qquad (9) \end{aligned}$$

For this minimization problem, the objective function and the only equality constraint (Equation (7)) are linear, the inequality constraints (Inequalities (8) and Inequalities (9)) are convex. Note that Inequalities (9) are convex because $1/f(x)$ is a convex function if $f(x)$ is a positive concave function, and $\sum_{j=1}^h z_j^{1/\alpha}$ is concave because $z_j^{1/\alpha}$ is concave for $\alpha > 1$. We have now transformed our min-max optimization problem into a convex minimization problem. We will prove that the following $(y^*, z_1^*, \cdots, z_P^*)$ is an optimal solution to the above convex minimization problem, by showing that it satisfies the KKT conditions.

$$y^* = \frac{h^{1-1/\alpha}}{\sum_{j=1}^{h}(z_j^*)^{1/\alpha}} \text{ for } h = 1, \cdots, P \tag{10}$$

Let $x_j = j^{1-1/\alpha} - (j-1)^{1-1/\alpha}$ for $j = 1, \cdots, P$, so that $x_j > x_{j+1}$. From Equation (10) and Equation (7), we get $z_j^* = b \cdot \frac{x_j}{\sum_{i=1}^{P} x_j^\alpha}$ and therefore $z_j^* > z_{j+1}^*$ for $j = 1, \cdots, P-1$.

To prove $(y^*, z_1^*, \cdots, z_P^*)$ satisfies the KKT conditions, we need to show that it satisfies primal feasibility, dual feasibility, complementary slackness, and stationarity. It is not hard to see that the proposed solution satisfies the primal feasibility in Equation (7), Inequalities (8) and Inequalities (9). Let us now associate multipliers with the constraints:

$$\begin{aligned} \lambda &: \sum_{j=1}^{P} z_j = b \\ w_j &: z_{j+1} - z_j \le 0 \text{ for } j = 1, ..., P-1 \\ \mu_h &: \frac{h^{1-1/\alpha}}{\sum_{j=1}^{h} z_j^{1/\alpha}} - y \le 0 \text{ for } h = 1, ..., P \end{aligned}$$

Since we have $z_j^* > z_{j+1}^*$ for $j = 1, \cdots, P-1$, then to satisfy complementary slackness, we have $w_j = 0$ for $j = 1, \cdots, P-1$. Now we need to show that there exists λ and $\mu_h \ge 0$ such that dual feasibility and stationarity are satisfied. To derive the stationarity condition, consider the Lagrangian function:

$$L(y, z_j, \lambda, \mu_h) = y + \sum_{h=1}^{P} \mu_h \left(\frac{h^{1-1/\alpha}}{\sum_{j=1}^{h} z_j^{1/\alpha}} - y \right) + \lambda \left(\sum_{j=1}^{P} z_j - b \right).$$

Taking the derivative of the Lagrangian function with respect to y and z_j, setting them to zero, and substituting in $(y^*, z_1^*, ..., z_P^*)$, we get the following set of stationarity conditions:

$$\sum_{h=1}^{P} \mu_h = 1, \tag{11}$$

$$\frac{(y^*)^2}{\alpha (z_j^*)^{1-1/\alpha}} \left(\sum_{h=j}^{P} \frac{\mu_h}{h^{1-1/\alpha}} \right) = \lambda \text{ for } j = 1, \cdots, P. \tag{12}$$

Solving the linear system in Equations (12) by considering μ_h as variables, we have $\mu_h = c_h \cdot \lambda$, where $c_h = \frac{h^{1-1/\alpha}\left((z_h^*)^{1-1/\alpha} - (z_{h+1}^*)^{1-1/\alpha} \right)\alpha}{(y^*)^2}$, for each $h = 1, \cdots, P$, and z_{P+1}^* is defined to be 0. According to the values of $(y^*, z_1^*, \cdots, z_P^*)$, we know that $y^* > 0$, $z_h^* > 0$ and $z_h^* > z_{h+1}^*$. Therefore, we have $c_h > 0$ for $h = 1, ..., P$. Substituting $\mu_h = c_h \cdot \lambda$ into Equation (11), we get $\lambda = \frac{1}{\sum_{h=1}^{P} c_h} > 0$, which implies that $\mu_h > 0$ for all $h = 1, \cdots, P$. Thus, we have shown that the dual feasibility is satisfied. Moreover, there exists λ and μ_h that make our proposed solution $(y^*, z_1^*, \cdots, z_P^*)$ satisfy stationarity, and hence all the KKT conditions. Therefore, it is an optimal solution for the convex minimization problem, and the corresponding speed assignment $s_j^* = (z_j^*)^{1/\alpha}$ is optimal for the original optimization problem. □

Using Lemma 7, the following theorem gives the lower bound for any non-clairvoyant algorithm.

Theorem 3 *Any non-clairvoyant algorithm is $\Omega(\ln^{1/\alpha} P)$-competitive with respect to total flow time plus energy, where P is the total number of processors.*

Proof. Consider a job set $\mathcal{J}$ containing only a single job with constant parallelism h and work w, where $1 \le h \le P$ and $w > 0$. For any non-clairvoyant algorithm A, we can assume without loss of generality that it allocates all P processors to the job with speeds $s_1 \ge s_2 \ge \cdots \ge s_P \ge 0$, which do not change throughout the job's execution since the work w can be arbitrarily small. The power consumption of A at any time is then given by $u = \sum_{j=1}^{P} s_j^{\alpha}$. The flow time plus energy of $\mathcal{J}$ scheduled by A is $G_A(\mathcal{J}) = (1+u)\frac{w}{\sum_{j=1}^{h} s_j}$. The optimal offline scheduler, knowing the parallelism h, will allocate exactly h processors of speed $\left(\frac{1}{(\alpha-1)h}\right)^{1/\alpha}$, thus incurring flow time plus energy of $G_{\mathsf{OPT}}(\mathcal{J}) = \frac{\alpha}{(\alpha-1)^{1-1/\alpha}} \cdot \frac{w}{h^{1-1/\alpha}}$. The competitive ratio of A is $\frac{G_{\mathsf{A}}(\mathcal{J})}{G_{\mathsf{OPT}}(\mathcal{J})} = \frac{(\alpha-1)^{1-1/\alpha}(1+u)}{\alpha} \cdot \frac{h^{1-1/\alpha}}{\sum_{j=1}^{h} s_j}$.

The adversary will choose parallelism h to maximize this ratio, i.e., to find $\max_{1\le h\le P} \frac{G_{\mathsf{A}}(\mathcal{J})}{G_{\mathsf{OPT}}(\mathcal{J})}$, while the online algorithm A chooses $(s_1, \cdots, s_P)$ to minimize $\max_{1\le h\le P} \frac{G_{\mathsf{A}}(\mathcal{J})}{G_{\mathsf{OPT}}(\mathcal{J})}$ regardless of the choice of h. According to Lemma 7, $\max_{1\le h\le P} \frac{h^{1-1/\alpha}}{\sum_{j=1}^{h} s_j}$ is minimized when $\frac{h^{1-1/\alpha}}{\sum_{j=1}^{h} s_j} = \frac{(h-1)^{1-1/\alpha}}{\sum_{j=1}^{h-1} s_j}$ for $h = 2, \cdots, P$. Hence, by solving this set of equations, the best non-clairvoyant algorithm will set $s_j = \left(j^{1-1/\alpha} - (j-1)^{1-1/\alpha}\right) s_1$ for $j = 1, 2, \cdots, P$. Since $j^{1-1/\alpha} - (j-1)^{1-1/\alpha} \ge \frac{1-1/\alpha}{j^{1/\alpha}}$, we have $s_j \ge \frac{1-1/\alpha}{j^{1/\alpha}} s_1$. Substituting these into $u = \sum_{j=1}^{P} s_j^{\alpha}$, we get $s_1 \le \frac{\alpha u^{1/\alpha}}{(\alpha-1)H_P^{1/\alpha}}$, where H_P is the P'th Harmonic number. The competitive ratio of any non-clairvoyant algorithm A thus satisfies $\frac{G_{\mathsf{A}}(\mathcal{J})}{G_{\mathsf{OPT}}(\mathcal{J})} \ge \frac{(\alpha-1)^{1-1/\alpha}(1+u)}{\alpha} \cdot \frac{1}{s_1} \ge \frac{(\alpha-1)^{2-1/\alpha}}{\alpha^2} \cdot \frac{1+u}{u^{1/\alpha}} H_P^{1/\alpha} \ge \frac{\alpha-1}{\alpha} \cdot H_P^{1/\alpha}$. The last inequality holds because $\frac{1+u}{u^{1/\alpha}}$ is minimized when $u = \frac{1}{\alpha-1}$. Since it is also known that $H_P = \Omega(\ln P)$, the theorem is proved. $\square$

3.4 IP-clairvoyant Algorithm: U-CEQ

In this section, we present an IP-clairvoyant algorithm called U-CEQ (Uniform Conservative Equi-Partitioning). We show that knowledge about the instantaneous parallelism of the jobs does help to improve the performance of an online algorithm. In particular, we prove that U-CEQ is $O(1)$-competitive with respect to total flow time plus energy, even for jobs with arbitrary release time.

Algorithm 2 describes the U-CEQ algorithm. As we can see, U-CEQ works similarly to N-EQUI, but it never allocates more processors to a job than the job's instantaneous parallelism at any time. Moreover, the speeds of the processors allocated to a job are assigned in a uniform manner.

To analyze the performance of U-CEQ, we say that job J_i is *satisfied* at time t if $a_i(t) = h_i^k$, and that it is *deprived* if $a_i(t) < h_i^k$. We see that at time t, a job J_i scheduled by U-CEQ has execution rate $\Gamma_i^k(t) = \frac{a_i(t)^{1-1/\alpha}}{(\alpha-1)^{1/\alpha}}$ and consumes power $u_i(t) = \frac{1}{\alpha-1}$. Therefore, the overall power consumption is given by $u_t = \frac{n_t}{\alpha-1}$. The following theorem gives the performance of U-CEQ for jobs with arbitrary released times.

Theorem 4 U-CEQ *is $\left(\max\{\frac{4\alpha^2}{\alpha-1}, 4^{\alpha}\alpha\} + 2\alpha\right)$-competitive with respect to total flow time plus energy for any set of parallel jobs.*

Algorithm 2 U-CEQ

```
Input: total number $P$ of processors, number $n_t$ of active jobs at time $t$ and the instantaneous parallelism $h_i^k$ of each active job $J_i$ at time $t$.
Output: number of allocated processors and their speeds for each active job at time $t$.
1: if $P \geq n_t$ then
2:     allocate $a_i(t) = \min\{h_i^k, \left\lfloor \frac{P}{n_t} \right\rfloor\}$ processors to each active job $J_i$.
3:     set the speed of all $a_i(t)$ processors allocated to job $J_i$ to be $s_i(t) = \left(\frac{1}{(\alpha-1)a_i(t)}\right)^{1/\alpha}$.
4: else
5:     allocate $a_i(t) = \frac{P}{n_t}$ fraction of a processor to each active job $J_i$.
6:     set the speed of all processors to be $s(t) = \left(\frac{n_t}{(\alpha-1)P}\right)^{1/\alpha}$.
7: end if
```

Proof. As with the analysis of N-EQUI, we prove the competitiveness of U-CEQ using amortized local competitiveness argument with the same potential function as in Equation (3), but with η now set to $\eta = \frac{\eta'}{P^{1-1/\alpha}}$, where $\eta' = \frac{4\alpha^2}{(\alpha-1)^{1-1/\alpha}}$. Clearly, the boundary, arrival and completion conditions continue to hold. We now show that the execution of any job set under U-CEQ (UC for short) satisfies the running condition $\frac{dG_{\mathsf{UC}}(\mathcal{J}(t))}{dt} + \frac{d\Phi(t)}{dt} \leq c_1 \cdot \frac{dG_{\mathsf{OPT}}(\mathcal{J}^*(t))}{dt} + c_2 \cdot \frac{dG_1^*(\mathcal{J}(t))}{dt}$, where $c_1 = \max\{\frac{4\alpha^2}{\alpha-1}, 4^\alpha\alpha\}$ and $c_2 = 2\alpha$.

Following the proof of Theorem 1, we have $\frac{dG_{\mathsf{UC}}(\mathcal{J}(t))}{dt} = \frac{\alpha}{\alpha-1}n_t$, $\frac{dG_{\mathsf{OPT}}(\mathcal{J}^*(t))}{dt} = n_t^* + \sum_{j=1}^{P}\left(s_j^*\right)^\alpha$, and $\frac{dG_1^*(\mathcal{J}(t))}{dt} \geq \frac{\alpha}{(\alpha-1)^{1-1/\alpha}}\sum_{J_i\in\mathcal{J}_S(t)}\frac{\Gamma_i^k(t)}{\left(h_i^k\right)^{1-1/\alpha}} = \frac{\alpha}{\alpha-1}(1-x_t)n_t$, where x_t is the deprived ratio as defined earlier. The rate of change $\frac{d\Phi(t)}{dt}$ for the potential function $\Phi(t)$ at time t can be shown to satisfy

$$\begin{aligned}
\frac{d\Phi(t)}{dt} &\leq \frac{\eta'}{P^{1-1/\alpha}}\left(-\sum_{i=1}^{n_t^D} i^{1-1/\alpha}\cdot\Gamma_i^k(t) + n_t^{1-1/\alpha}\sum_{j=1}^{P}s_j^* + n_t^*\sum_{i=1}^{n_t}\left(i^{1-1/\alpha}-(i-1)^{1-1/\alpha}\right)\Gamma_i^k(t)\right)\\
&\leq \eta'\left(-\frac{x_t^2}{4(\alpha-1)^{1/\alpha}}n_t + \frac{\lambda}{\alpha}\sum_{j=1}^{P}\left(s_j^*\right)^\alpha + \frac{1-1/\alpha}{\lambda^{1/(\alpha-1)}}n_t + \frac{n_t^*}{(\alpha-1)^{1/\alpha}}\right),
\end{aligned}$$

where $\lambda = 4^{\alpha-1}(\alpha-1)^{1-1/\alpha}$. Substituting these bounds into the running condition, we see that it is satisfied for all valid values of x_t. Hence, the theorem is proved. $\square$

Theorem 4 shows that U-CEQ is $O(1)$-competitive for total flow time plus energy, since α can be considered as a constant with respect to P. The competitive ratio, however, is still exponential in α. The following theorem shows that, for jobs with identical release time, the competitive ratio can be further improved to be strictly smaller than 6 regardless of the value of α.

Theorem 5 U-CEQ *is* $\left(2^{2-1/\alpha}+2\right)$*-competitive with respect to total flow time plus energy for any set of parallel jobs with identical release time.*

Proof. Similarly to the proof for N-EQUI, we prove the competitiveness of U-CEQ for jobs with identical release time using the local competitive argument. In particular, we show that any job set scheduled by U-CEQ satisfies the running condition $\frac{dG_{\mathsf{UC}}(\mathcal{J}(t))}{dt} \leq$

$2 \cdot \frac{dG_1^*(\mathcal{J}^*(t))}{dt} + 2^{2-1/\alpha} \cdot \frac{dG_2^*(\mathcal{J}(t))}{dt}$. Again, we have $\frac{dG_{\mathsf{UC}}(\mathcal{J}(t))}{dt} = \frac{\alpha}{\alpha-1}n_t$ and $\frac{dG_1^*(\mathcal{J}(t))}{dt} \geq \frac{\alpha}{\alpha-1}(1-x_t)n_t$ from the proof of Theorem 4. Also, by following the proof of Theorem 2, we can get $\frac{dG_2^*(\mathcal{J}(t))}{dt} \geq \frac{\alpha}{((\alpha-1)P)^{1-1/\alpha}} \sum_{i=1}^{n_t^D} i^{1-1/\alpha} \cdot \Gamma_i^k(t) \geq \frac{\alpha}{\alpha-1} \cdot \frac{x_t^2 n_t}{2^{2-1/\alpha}}$. Using these bounds, the desired running condition can be satisfied for all valid values of x_t. □

U-CEQ significantly improves upon any non-clairvoyant algorithm with respect to total flow time plus energy. This is essentially due to U-CEQ not wasting any energy yet still guaranteeing a sufficient execution rate for the jobs. Since it is known that non-clairvoyant algorithms perform similarly to the IP-clairvoyant ones with respect to total flow time alone [21, 23, 33, 24], Theorems 4 and 5 show the importance of even partial clairvoyance when energy is also of concern. Moreover, since U-CEQ is aware of the instantaneous parallelism of the jobs, uniform speed scaling is sufficient to ensure its competitiveness. Therefore, compared to the non-clairvoyant algorithm N-EQUI, which requires non-uniform speed scaling, U-CEQ may be more feasible in practice, especially when scheduling for jobs whose parallelism does not change frequently. Lastly, note that non-uniform speed scaling is not beneficial in the IP-clairvoyant setting. Indeed, using non-uniform speeds can only degrade an algorithm's performance, since generally less energy will be consumed at the same execution rate when using a uniform speed [49].

4 Makespan Plus Energy

In this section, we consider the objective of minimizing makespan plus energy. In particular, we propose an IP-clairvoyant algorithm called WCEP, and show that it is $O(\ln^{1-1/\alpha} P)$-competitive for any set of parallel jobs regardless of their release time. We also show that this ratio is asymptotically optimal for any IP-clairvoyant algorithm.

4.1 Performance of the Optimal

To bound the performance of WCEP, we first derive a bound on the performance of the optimal offline scheduler. It turns out that, for makespan plus energy, we need only focus on the case where all jobs are released together, as we will show in Section 4.2.2, adding release time to the jobs will increase the competitive ratio of our algorithm by at most a constant factor in the worst case.

In the following lemma, we show that to minimize makespan plus energy for jobs with identical release time, the optimal scheduler always maintains a constant total power of $\frac{1}{\alpha-1}$ at any time. A similar property was proven in [41] for the makespan minimization problem with a total energy budget.

Lemma 8 *Given a set of jobs with identical release time, the optimal scheduler will execute the jobs with a constant total power of* $\frac{1}{\alpha-1}$ *at any time during the execution.*

Proof. We prove the lemma by contradiction.

Consider an interval Δt during which the speeds of all processors, denoted by $(s_1, s_2, \cdots, s_P)$, remain unchanged in the optimal schedule. The makespan plus energy incurred when executing this portion of the job set is given by $H = \Delta t(1+u)$, where $u = \sum_{j=1}^{P} s_j^\alpha$ is the power consumption of all the processors during Δt. Suppose that $u \neq \frac{1}{\alpha-1}$. We will show that by modifying the power consumption, we can reduce the overall makespan plus energy.

Specifically, the modified schedule executes the same portion of the job set by running the j'th processor at speed $k \cdot s_j$, where $k = \left(\frac{1}{(\alpha-1)u}\right)^{1/\alpha}$. This portion will then finish in $\frac{\Delta t}{k}$ time, and consumes $\frac{1}{\alpha-1}$ power at any time during this interval. The new makespan plus energy incurred when executing this portion of the job set is $H' = \frac{\Delta t}{k}(1+\frac{1}{\alpha-1}) = \frac{\alpha}{(\alpha-1)^{1-1/\alpha}}\Delta t u^{1/\alpha}$. Hence, we have $\frac{H}{H'} = \frac{(\alpha-1)^{1-1/\alpha}}{\alpha} \cdot \frac{1+u}{u^{1/\alpha}} > 1$, i.e., $H > H'$, since $\frac{1+x}{x^{1/\alpha}}$ is uniquely minimized at $x = \frac{1}{\alpha-1}$ for all $x > 0$ and $u \neq \frac{1}{\alpha-1}$. While the costs incurred when executing other portions of the job set are unchanged, the modified schedule incurs a strictly less makespan plus energy. This contradicts the fact that the original schedule is optimal. □

Intuitively, since the jobs contribute 1 towards the makespan part of the objective function at any time during their execution, Lemma 8 implies that the optimal strategy provides a balanced contribution towards the power consumption part of the objective at all time.[6] However, this result only holds when the jobs have identical release time. For jobs with arbitrary release time, speeding up the execution whenever the power is less than $\frac{1}{\alpha-1}$ may not be helpful, for the subsequent jobs may not have arrived yet to fill in the gap. Therefore, the optimal power consumption in this case would be upper-bounded by $\frac{1}{\alpha-1}$.

Lemma 8 shows that the total power consumed by all the jobs should be constant over time in the optimal schedule, provided that all jobs are released at the same time. The following lemma shows that the optimal scheduler also uses a constant power throughout the execution of any individual job, provided that sufficient processors are available.

Lemma 9 *Given a set of jobs, suppose that there are sufficient processors available to satisfy all jobs at all times, i.e.,* $P \geq \sum_{i=1..n} h_i^{max}$, *where* $h_i^{max} = \max_{k=1..k_i} h_i^k$. *Then, the optimal scheduler will allocate a constant power to any individual job throughout its execution lifetime.*

Proof. Again, we prove the lemma by contradiction.

Since there are sufficient processors in the system, by the convexity of the power function, the optimal scheduler should allocate exactly h_i^k processors of the same speed to each phase J_i^k. Now, suppose that there exist two phases from the same job, to which the optimal scheduler does not allocate the same power, and let $\langle w_1, h_1 \rangle$ and $\langle w_2, h_2 \rangle$ denote the work-parallelism pairs of these two phases, respectively. Thus, we have $h_1 s_1^\alpha \neq h_2 s_2^\alpha$, where s_1 and s_2 denote the speeds of the processors allocated to the two phases. We will show that, by modifying the power allocations for the two phases, we can reduce the overall energy consumption while maintaining the execution time of the job.

Let $t_1 = \frac{w_1}{h_1 s_1}$ and $t_2 = \frac{w_2}{h_2 s_2}$. The overall execution time and energy consumption of the two phases are given by $T = t_1 + t_2$ and $E = t_1 \cdot h_1 s_1^\alpha + t_2 \cdot h_2 s_2^\alpha$, respectively. Let s_1' and s_2' denote the speeds used for the two phases in the modified schedule, and we will make sure that their power consumptions are identical, i.e.,

$$h_1(s_1')^\alpha = h_2(s_2')^\alpha = u. \tag{13}$$

Moreover, to maintain the same execution time for the job, the processor speeds in the modified schedule should also satisfy $\frac{w_1}{h_1 s_1'} + \frac{w_2}{h_2 s_2'} = t_1 + t_2$. This gives us

$$1 = \beta \cdot \frac{s_1}{s_1'} + (1-\beta) \cdot \frac{s_2}{s_2'}, \tag{14}$$

[6]A similar argument will show that the optimal strategy for the objective of total flow time plus energy maintains a power consumption of $\frac{n_t}{\alpha-1}$ at any time t, where n_t is the number of active jobs at time t. Lemma 1 shows that the optimal scheduler indeed satisfies this property.

where $\beta = \frac{t_1}{t_1+t_2}$. Solving Equations (13) and (14), we get $u = \left(\beta \cdot h_1^{1/\alpha} s_1 + (1-\beta) \cdot h_2^{1/\alpha} s_2\right)^\alpha$, and $s'_1 = \left(\frac{u}{h_1}\right)^{1/\alpha}$ and $s'_2 = \left(\frac{u}{h_2}\right)^{1/\alpha}$. The total energy consumption for the two phases in the modified schedule is then given by

$$\begin{aligned}
E' &= \left(\frac{w_1}{h_1 s'_1} + \frac{w_2}{h_2 s'_2}\right) u \\
&= \left(t_1 \cdot \frac{s_1}{s'_1} + t_2 \cdot \frac{s_2}{s'_2}\right) u \\
&= \left(t_1 \cdot h_1^{1/\alpha} s_1 + t_2 \cdot h_2^{1/\alpha} s_2\right) u^{1-1/\alpha} \\
&= (t_1 + t_2) \left(\beta \cdot h_1^{1/\alpha} s_1 + (1-\beta) \cdot h_2^{1/\alpha} s_2\right)^\alpha \\
&< (t_1 + t_2) \left(\beta \cdot h_1 s_1^\alpha + (1-\beta) \cdot h_2 s_2^\alpha\right) \\
&= t_1 \cdot h_1 s_1^\alpha + t_2 \cdot h_2 s_2^\alpha \\
&= E.
\end{aligned}$$

The inequality is because x^α is strictly convex for $\alpha > 1$, and $h_1^{1/\alpha} s_1 \neq h_2^{1/\alpha} s_2$ according to our assumption. Hence, the modified schedule consumes strictly less energy while having the same makespan. This contradicts the fact that the original schedule is optimal. □

We now give lower bounds on the performance of the optimal offline scheduler. In contrast to the total flow time plus energy, where the completion time of each job contributes to the overall objective function, the makespan for a set of jobs only depends on the completion time of the last job. Hence, the other jobs only contribute to the energy consumption part of the objective, and can therefore be slowed down to consume less energy and improve the overall performance. Based on this observation as well as Lemmas 8 and 9, we derive the following two lower bounds.

Lemma 10 *The optimal makespan plus energy for any set $\mathcal{J}$ of jobs with identical release time satisfies $H_{\mathsf{OPT}}(\mathcal{J}) \geq \max\{H_1^*(\mathcal{J}), H_2^*(\mathcal{J})\}$, where*

$$H_1^*(\mathcal{J}) = \frac{\alpha}{(\alpha-1)^{1-1/\alpha}} \cdot \frac{\sum_{i=1}^n w(J_i)}{P^{1-1/\alpha}}, \tag{15}$$

$$H_2^*(\mathcal{J}) = \frac{\alpha}{(\alpha-1)^{1-1/\alpha}} \cdot \left(\sum_{i=1}^n x(J_i)^\alpha\right)^{1/\alpha}, \tag{16}$$

and where $w(J_i)$ and $x(J_i)$ denote the work and the unit-time span of job J_i respectively, and P is the total number of processors.

Proof. By Lemma 8, the optimal scheduler at any time t consumes power $u_t^* = \frac{1}{\alpha-1}$. Hence, the energy consumption $E_{\mathsf{OPT}}(\mathcal{J})$ under the optimal schedule satisfies $E_{\mathsf{OPT}}(\mathcal{J}) = \int_0^{M_{\mathsf{OPT}}(\mathcal{J})} u_t^* dt = \frac{1}{\alpha-1} M_{\mathsf{OPT}}(\mathcal{J})$. The optimal makespan plus energy is then given by $H_{\mathsf{OPT}}(\mathcal{J}) = M_{\mathsf{OPT}}(\mathcal{J}) + E_{\mathsf{OPT}}(\mathcal{J}) = \frac{\alpha}{\alpha-1} M_{\mathsf{OPT}}(\mathcal{J})$. Therefore, we only focus on the makespan in the following.

To get the first lower bound, we observe that the maximum execution rate on P processors using a total power of $\frac{1}{\alpha-1}$ is achieved when all processors run at the same speed of $s = \left(\frac{1}{(\alpha-1)P}\right)^{1/\alpha}$. Since the total work of all jobs in the job set is $\sum_{i=1}^n w(J_i)$, the optimal makespan satisfies $M_{\mathsf{OPT}}(\mathcal{J}) \geq \frac{\sum_{i=1}^n w(J_i)}{Ps} = (\alpha-1)^{1/\alpha} \cdot \frac{\sum_{i=1}^n w(J_i)}{P^{1-1/\alpha}}$.

For the second lower bound, we give infinite number of processors to the optimal scheduler, which can only improve its performance. Thus, all jobs can be satisfied at all times, and the optimal scheduler will allocate exactly h_i^k processors of the same speed s_i^k to each phase J_i^k. By Lemma 9, the power consumption of any job is constant over all phases, i.e., for each job $J_i \in \mathcal{J}$, we have $h_i^1(s_i^1)^\alpha = h_i^k(s_i^k)^\alpha$ for all $1 \le k \le k_i$. Hence, the execution rate for phase J_i^k is given by $h_i^k s_i^k = (h_i^k)^{1-1/\alpha}(h_i^1)^{1/\alpha}s_i^1$, and so the completion time of job J_i satisfies $c_i = \sum_{k=1}^{k_i} \frac{w_i^k}{h_i^k s_i^k} = \frac{x(J_i)}{(h_i^1)^{1/\alpha}s_i^1}$. Moreover, the optimal scheduler will finish all the jobs at the same time, since otherwise slowing down the jobs that finish early will reduce the energy without increasing makespan. This gives $\frac{x(J_1)}{(h_1^1)^{1/\alpha}s_1^1} = \frac{x(J_i)}{(h_i^1)^{1/\alpha}s_i^1}$ for all $1 \le i \le n$. By Lemma 8, the total power is constant when the optimal scheduler starts to execute the first phase of all jobs, i.e., $\sum_{i=1}^n h_i^1(s_i^1)^\alpha = \frac{1}{\alpha-1}$. Solving all these equations gives us $s_i^1 = \left(\frac{1}{\alpha-1}\right)^{1/\alpha} \cdot \frac{x(J_i)}{(h_i^1)^{1/\alpha}} \cdot \frac{1}{\left(\sum_{i=1}^n x(J_i)^\alpha\right)^{1/\alpha}}$. The optimal makespan is thus $M_{\mathsf{OPT}}(\mathcal{J}) = c_i = \frac{x(J_i)}{(h_i^1)^{1/\alpha}s_i^1} = (\alpha-1)^{1/\alpha} \cdot \left(\sum_{i=1}^n x(J_i)^\alpha\right)^{1/\alpha}$. □

4.2 IP-clairvoyant Algorithm: WCEP

We now present an IP-clairvoyant algorithm WCEP (Work-Conserving Equal-Power) for any set of parallel jobs. Algorithm 3 describes its details.

There are two operating modes in WCEP, namely Work-Conserving (WC) and Equal-Power (EP). Whenever the sum of the instantaneous parallelism of all active jobs exceeds the total number of processors, the algorithm enters WC mode. In this mode, the P processors can be allocated in any manner as long as all processors are assigned and no job receives more processors than its instantaneous parallelism. This can be achieved by any work-conserving algorithm, such as Proportional Allocation [14] or Dynamic Equi-partitioning [13, 21]. All processors in this mode share the same speed. On the other hand, when the total instantaneous parallelism of all active jobs is not more than the total number of processors, the algorithm enters EP mode, in which each job receives exactly the same number of processors as its instantaneous parallelism and all jobs consumes the same power.

Algorithm 3 WCEP

```
Input: total number $P$ of processors and the instantaneous parallelism $h_i^k$ of each active
    job $J_i \in \mathcal{J}(t)$ at time $t$.
Output: number of allocated processors and their speeds for each active job at time $t$.
1: if $\sum_{J_i \in \mathcal{J}(t)} h_i^k > P$ then
2:     allocate $a_i(t) \le h_i^k$ processors to each job $J_i \in \mathcal{J}(t)$ subject to $\sum_{J_i \in \mathcal{J}(t)} a_i(t) = P$.
3:     set the speed of all $P$ processors to be $s(t) = \left(\frac{1}{(\alpha-1)P}\right)^{1/\alpha}$.
4: else
5:     allocate $a_i(t) = h_i^k$ processors to each job $J_i \in \mathcal{J}(t)$;
6:     set the speed of all $a_i(t)$ processors allocated to job $J_i$ to be $s_i(t) = \left(\frac{1}{(\alpha-1)a_i(t)n_t}\right)^{1/\alpha}$.
7: end if
```

Similarly to the optimal offline scheduler, we can see from Algorithm 3 that WCEP consumes a total power of $u_t = \frac{1}{\alpha-1}$ at any time t. Hence, the overall energy consumption satisfies $E(\mathcal{J}) = \frac{1}{\alpha-1}M(\mathcal{J})$, and the makespan plus energy is given by $H(\mathcal{J}) = \frac{\alpha}{\alpha-1}M(\mathcal{J})$.

For convenience, we drop the algorithm subscript in this section and let $H(\mathcal{J})$ denote $H_{\mathsf{WCEP}}(\mathcal{J})$, since WCEP is the only algorithm we study for this objective.

4.2.1 Performance for Jobs with Identical Release Time

Before analyzing the performance of WCEP for jobs with identical release time, we first define some notations and prove a useful lemma. For each job J_i, let J_i^{wc} and J_i^{ep} denote the portions of the job executed under WC and EP modes, respectively. Moreover, we define $\mathcal{J}^{wc} = \{J_i^{wc} : J_i \in \mathcal{J}\}$ and $\mathcal{J}^{ep} = \{J_i^{ep} : J_i \in \mathcal{J}\}$. The following lemma shows that all jobs in $\mathcal{J}^{ep}$ reduce their unit-power span at the same rate.

Lemma 11 *Suppose that* WCEP *schedules a set of jobs with identical release time. Then at any time t in EP mode, the unit-power span of any active job in $\mathcal{J}^{ep}$ is reduced at the rate of $\left(\frac{1}{(\alpha-1)n_t}\right)^{1/\alpha}$ regardless of its instantaneous parallelism, where n_t is the total number of active jobs at time t.*

Proof. At any time t when WCEP is in EP mode, let h_i^k denote the instantaneous parallelism of active job $J_i \in \mathcal{J}^{ep}$. According to Algorithm 3, the work of the job is reduced at a rate of $\frac{dw(J_i^{ep})}{dt} = a_i(t)s_i(t) = \frac{(h_i^k)^{1-1/\alpha}}{((\alpha-1)n_t)^{1/\alpha}}$. Hence, by definition, the unit-power span of the job is reduced at a rate of $\frac{dx(J_i^{ep})}{dt} = \frac{dw(J_i^{ep})}{dt} \cdot \frac{1}{(h_i^k)^{1-1/\alpha}} = \left(\frac{1}{(\alpha-1)n_t}\right)^{1/\alpha}$. □

The following theorem shows the competitive ratio of WCEP for jobs with identical release time.

Theorem 6 WCEP *is $\Theta(\ln^{1-1/\alpha} P)$-competitive with respect to makespan plus energy for any set of jobs with identical release time, where P is the total number of processors.*

Proof. We again only focus on the makespan $M(\mathcal{J})$ of the job set scheduled by WCEP, since the makespan plus energy satisfies $H(\mathcal{J}) = \frac{\alpha}{\alpha-1}M(\mathcal{J})$. We separately bound the time $M_{wc}(\mathcal{J})$ when the algorithm is in WC mode and the time $M_{ep}(\mathcal{J})$ when the algorithm is in EP mode. Obviously, we have $M(\mathcal{J}) = M_{wc}(\mathcal{J}) + M_{ep}(\mathcal{J})$.

We first bound $M_{wc}(\mathcal{J})$. According to WCEP, the total execution rate for the active jobs at any time t in WC mode is given by $\frac{P^{1-1/\alpha}}{(\alpha-1)^{1/\alpha}}$. Since the total work of all jobs in $\mathcal{J}^{wc}$ satisfies $\sum_{i=1}^n w(J_i^{wc}) \leq \sum_{i=1}^n w(J_i)$, we have $M_{wc}(\mathcal{J}) \leq (\alpha-1)^{1/\alpha}\frac{\sum_{i=1}^n w(J_i)}{P^{1-1/\alpha}}$.

We now bound $M_{ep}(\mathcal{J})$ when the algorithm is in EP mode. Let T denote the first time instance when the algorithm enters EP mode, and let m denote the number of active jobs at T, i.e., $m = n_T$. Since the instantaneous parallelism of each active job is at least 1 and all m active jobs are satisfied at T, we have $m < P$. For convenience, rename the jobs in $\mathcal{J}^{ep}$ in non-decreasing order of their unit-power span, i.e., $x(J_1^{ep}) \leq x(J_2^{ep}) \leq \cdots \leq x(J_m^{ep})$. According to Lemma 11, whenever the algorithm is in EP mode, WCEP will reduce the unit-power span of all the active jobs in $\mathcal{J}^{ep}$ at the same rate of $\left(\frac{1}{(\alpha-1)n_t}\right)^{1/\alpha}$. Thus, the jobs in $\mathcal{J}^{ep}$ will complete in exactly the above order. Let $x(J_0^{ep}) = 0$. Then we have $M^{ep}(\mathcal{J}) = \sum_{i=1}^m \frac{x(J_i^{ep})-x(J_{i-1}^{ep})}{\left(\frac{1}{(\alpha-1)(m-i+1)}\right)^{1/\alpha}} = (\alpha-1)^{1/\alpha}\sum_{i=1}^m \left((m-i+1)^{1/\alpha} - (m-i)^{1/\alpha}\right)x(J_i^{ep})$. For convenience, define $c_i = (m-i+1)^{1/\alpha} - (m-i)^{1/\alpha}$ for $1 \leq i \leq m$, so we have

$c_i \le \frac{1}{(m-i+1)^{1-1/\alpha}}$. Let $R = \sum_{i=1}^{m} x(J_i^{ep})^\alpha$, and subject to this condition and the ordering of $x(J_i^{ep})$, we can show using Lagrange multipliers that $\sum_{i=1}^{m} c_i \cdot x(J_i^{ep})$ is maximized when $x(J_i^{ep}) = R^{1/\alpha} \cdot c_i^{\frac{1}{\alpha-1}} / \left(\sum_{i=1}^{m} c_i^{\frac{\alpha}{\alpha-1}}\right)^{1/\alpha}$. Hence, we have $M^{ep}(\mathcal{J}) \le (\alpha-1)^{1/\alpha} R^{1/\alpha} \left(\sum_{i=1}^{m} c_i^{\frac{\alpha}{\alpha-1}}\right)^{1-1/\alpha} \le (\alpha-1)^{1/\alpha} R^{1/\alpha} H_m^{1-1/\alpha}$, where $H_m = 1 + 1/2 + \cdots + 1/m$ is the m'th harmonic number.

The makespan plus energy of the job set scheduled under WCEP thus satisfies $H(\mathcal{J}) \le \frac{\alpha}{(\alpha-1)^{1-1/\alpha}} \cdot \left(\frac{\sum_{i=1}^{n} w(J_i)}{P^{1-1/\alpha}} + R^{1/\alpha} H_m^{1-1/\alpha}\right)$. Thus, since $\sum_{i=1}^{n} x(J_i)^\alpha \ge \sum_{i=1}^{m} x(J_i^{ep})^\alpha = R$, we get using Lemma 10 that $H(\mathcal{J}) \le (1 + H_m^{1-1/\alpha}) \cdot H_{\mathsf{OPT}}(\mathcal{J}) = O(\ln^{1-1/\alpha} P) \cdot H_{\mathsf{OPT}}(\mathcal{J})$.

To show that this ratio is asymptotically optimal for WCEP, consider a set $\mathcal{J}$ of P sequential jobs released at time 0, where the i'th job has unit-power span $x(J_i) = \frac{1}{(P-i+1)^{1/\alpha}}$. From Lemma 10, the optimal scheduler has makespan plus energy $H_{\mathsf{OPT}}(\mathcal{J}) = \frac{\alpha}{(\alpha-1)^{1-1/\alpha}} H_P^{1/\alpha}$, where H_P is the P'th harmonic number. From the above proof, the performance of WCEP is given by $H(\mathcal{J}) = \frac{\alpha}{\alpha-1} M(\mathcal{J}) = \frac{\alpha}{(\alpha-1)^{1-1/\alpha}} \sum_{i=1}^{P} \left((P-i+1)^{1/\alpha} - (P-i)^{1/\alpha}\right) x(J_i) \ge \frac{\alpha}{(\alpha-1)^{1-1/\alpha}} \sum_{i=1}^{P} \frac{x(J_i)}{\alpha(P-i+1)^{1-1/\alpha}} = \frac{1}{(\alpha-1)^{1-1/\alpha}} H_P$. The competitive ratio of WCEP in this case is thus $\frac{H(\mathcal{J})}{H_{\mathsf{OPT}}(\mathcal{J})} \ge \frac{1}{\alpha} \cdot H_P^{1-1/\alpha} = \Omega(\ln^{1-1/\alpha} P)$. □

4.2.2 Performance for Jobs with Arbitrary Release Time

In this section, we show that WCEP has the same asymptotic performance with respect to makespan plus energy when the jobs can have arbitrary release time. In fact, the competitive ratio as compared to the case with identical release time will increase by an additive factor of $\frac{\alpha}{\alpha-1}$ in the worst case.

For convenience, we assume that the jobs in any job set $\mathcal{J}$ are renamed according to their release time, i.e., $0 = r_1 \le r_2 \le \cdots \le r_n$. Obviously, the last release time r_n is a lower bound on the makespan plus energy of job set $\mathcal{J}$, i.e., $H_{\mathsf{OPT}}(\mathcal{J}) \ge r_n$. Moreover, the two lower bounds shown in Lemma 10 will continue to hold even though Lemma 8 cannot be applied to jobs with arbitrary release time. To see this, define a corresponding job set $\mathcal{J}'$, which contains exactly the same set of jobs in $\mathcal{J}$ but with the release time of all jobs set to 0. The optimal schedule for $\mathcal{J}$ is a valid schedule for $\mathcal{J}'$, which implies $H_{\mathsf{OPT}}(\mathcal{J}) \ge H_{\mathsf{OPT}}(\mathcal{J}')$. Furthermore, the corresponding jobs in $\mathcal{J}$ and $\mathcal{J}'$ share the same work and unit-power span, which gives $H_1^*(\mathcal{J}) = H_1^*(\mathcal{J}')$ and $H_2^*(\mathcal{J}) = H_2^*(\mathcal{J}')$ according to definition. Hence, we have $H_{\mathsf{OPT}}(\mathcal{J}) \ge H_{\mathsf{OPT}}(\mathcal{J}') \ge \max\{H_1^*(\mathcal{J}'), H_2^*(\mathcal{J}')\} = \max\{H_1^*(\mathcal{J}), H_2^*(\mathcal{J})\}$, where the second inequality follows by Lemma 10.

The following theorem gives the performance of WCEP for the general case.

Theorem 7 WCEP *is* $\Theta(\ln^{1-1/\alpha} P)$*-competitive with respect to makespan plus energy for any set of parallel jobs, where* P *is the total number of processors.*

Proof. Using the notions of t-prefix and t-suffix introduced in Section 3.1.2, we define $\mathcal{J}(\overleftarrow{r_n})$ to be the r_n-prefix of the job set $\mathcal{J}$, or the portion of the job set completed before and on time r_n, under the schedule of WCEP. Similarly, we define $\mathcal{J}(\overrightarrow{r_n})$ to be the r_n-suffix of the job set $\mathcal{J}$ scheduled under WCEP. Since all jobs in $\mathcal{J}$ have arrived by time r_n, the makespan plus energy incurred by WCEP for the entire job set $\mathcal{J}$ is given by $H(\mathcal{J}) = H(\mathcal{J}(\overleftarrow{r_n})) + H(\mathcal{J}(\overrightarrow{r_n}))$.

According to the definition of WCEP, the makespan plus energy incurred by executing $\mathcal{J}(\overleftarrow{r_n})$ is $H(\mathcal{J}(\overleftarrow{r_n})) = \frac{\alpha}{\alpha-1}M(\mathcal{J}(\overleftarrow{r_n})) = \frac{\alpha}{\alpha-1}r_n$. By the proof of Theorem 6, the makespan plus energy incurred by executing $\mathcal{J}(\overrightarrow{r_n})$ is $H(\mathcal{J}(\overrightarrow{r_n})) \le H_1^*(\mathcal{J}(\overrightarrow{r_n})) + H_P^{1-1/\alpha} \cdot H_2^*(\mathcal{J}(\overrightarrow{r_n}))$, where H_P is the P'th harmonic number. Apparently, we have $H_1^*(\mathcal{J}(\overrightarrow{r_n})) \le H_1^*(\mathcal{J})$ and $H_2^*(\mathcal{J}(\overrightarrow{r_n})) \le H_2^*(\mathcal{J})$ by the definitions of H_1^* and H_2^*. Thus, based on the three lower bounds for the makespan plus energy, the total cost of WCEP satisfies $H(\mathcal{J}) \le \left(\frac{\alpha}{\alpha-1} + 1 + H_P^{1-1/\alpha}\right) H_{\mathsf{OPT}}(\mathcal{J}) = O(\ln^{1-1/\alpha} P) \cdot H_{\mathsf{OPT}}(\mathcal{J})$. □

From the proofs of Theorems 6 and 7, we can see that the cost of the WCEP algorithm when executing in WC mode can be amortized against the cost of the optimal offline scheduler. Hence, the competitive ratio of WCEP comes primarily from the execution of the jobs in EP mode, during which sufficient processors are available. The strategy of WCEP in this mode is to give each active job the same amount of power, thus reducing the jobs' unit-power span at the same rate. Without knowing the jobs' remaining characteristics, this strategy seems to provide an optimal solution for any online algorithm. In the next section, we confirm the intuition by proving a matching lower bound for any IP-clairvoyant algorithm, which shows that WCEP is asymptotically optimal with respect to makespan plus energy.

4.3 Lower Bound for Any IP-clairvoyant Algorithm

In this section, we present an $\Omega(\ln^{1-1/\alpha} P)$ lower bound on the competitiveness of any IP-clairvoyant algorithm with respect to makespan plus energy. The idea is to show that, without any knowledge about the remaining characteristics of the jobs, the WCEP algorithm will perform no worse than any IP-clairvoyant algorithm under a particular adversarial strategy. This is achieved by transforming any IP-clairvoyant schedule for a set of sequential jobs into a WCEP schedule without increasing the overall cost.

Before proving the lower bound, we first consider the following scenario, which represents an intermediate state of the jobs during the transformation process.

Scenario 1 *Suppose that there are two sequential jobs with the same total work. Each job is divided into m segments, where $m \ge 1$, and the corresponding segments of the two jobs also have the same work. Each job is to be executed independently on a processor, starting at time 0 and ending at time T, where $T > 0$. Each segment of a job is to be executed with the same speed, but different segments of the same job or the corresponding segments of different jobs may be executed with different speeds. Let A denote any valid schedule in this scenario. For each job J_i scheduled by A, where $i = 1, 2$, let t_i^j denote the completion time for the j'th segment of the job, where $1 \le j \le m$. Hence, we have $0 < t_i^1 < t_i^2 < \cdots < t_i^m = T$.*

For the scenario described above, the following lemma transforms any valid schedule A that executes the two jobs differently into a more energy-efficient schedule that executes the two jobs identically throughout execution.

Lemma 12 *For any valid schedule A satisfying Scenario 1, there exists a valid schedule B that executes the two jobs identically, i.e., using the same speed at any time during their execution, and which consumes no more energy than A. Moreover, the time t^j when the j'th segment of both jobs is completed in B satisfies $\min\{t_1^j, t_2^j\} \le t^j \le \max\{t_1^j, t_2^j\}$ for each $1 \le j \le m-1$.*

Proof. We prove the lemma by induction on the number m of segments.

In the base case, we have $m = 1$. Since both jobs are started and completed at the same time, their execution speeds are identical. Hence, the claim holds trivially.

For the inductive step, let $m \geq 1$ and suppose that the claim holds when the jobs consist of K segments for each $1 \leq K \leq m$. We will show that the claim also holds when the jobs have $K = m + 1$ segments. For convenience, let $t_1^0 = t_2^0 = 0$ and let $t_1^{m+1} = t_2^{m+1} = T$. Without loss of generality, we can assume $t_1^1 \leq t_2^1$ under schedule A. Let l denote the smallest index that satisfies $t_1^l \geq t_2^l$ and $l \geq 1$. Note that $l = 1$ if $t_1^1 = t_2^1$, and $l = m + 1$ if $t_1^j < t_2^j$ for all $1 \leq j \leq m$. Also, let s_i^j denote the execution speed for the j'th segment of job J_i. Our goal is to transform schedule A by adjusting the speeds s_1^1, s_2^1, s_1^l and s_2^l to achieve $t_1^j = t_2^j$ for some $1 \leq j \leq l - 1$, while not increasing the total energy consumption. Then, the adjusted time t_1^j (or t_2^j) divides each job J_i into two parts J_i' and J_i'' with j and $m + 1 - j$ segments, respectively. By the inductive hypothesis, there exists a more energy-efficient schedule B' that executes J_1' and J_2' identically in $[0, t_1^j]$, and similarly there is a more energy-efficient schedule B'' that executes J_1'' and J_2'' identically in $[t_1^j, T]$. Schedule B is then obtained by combining B' and B''. Now, to achieve $t_1^j = t_2^j$ for some $1 \leq j \leq l - 1$, we distinguish two cases.

Case 1: $s_1^1 \leq s_1^l$. Since we assumed that l is the smallest index to satisfy $t_1^l \geq t_2^l$, we have $t_1^{l-1} < t_2^{l-1}$. As we also assumed $t_1^1 \leq t_2^1$, we can observe that $s_2^1 \leq s_1^1 \leq s_1^l \leq s_2^l$. In this case, we can decrease t_2^j for each $1 \leq j \leq l-1$ by an infinitesimal amount of time Δt by increasing speed s_2^1 and reducing speed s_2^l, while keeping $s_2^1 \leq s_2^l$. By the convexity of the power function, the total energy consumption will not increase. Repeat this process until we get $t_1^j = t_2^j$ for some $1 \leq j \leq l-1$, which is always possible due to the above observation.

Case 2: $s_1^1 > s_1^l$. In this case, we can increase t_1^j for each $1 \leq j \leq l-1$ by an infinitesimal amount of time Δt by reducing speed s_1^1 and increasing speed s_1^l. Again, the total energy consumption will not increase, by the convexity of the power function. Repeat this process until we get $t_1^j = t_2^j$ for some $1 \leq j \leq l - 1$ or $s_1^1 = s_1^l$. In the latter case, the situation can be handled by Case 1.

Observe that the speed adjustments in both cases make each pair of time instances t_1^j and t_2^j, for any $1 \leq j \leq l - 1$, shift toward each other. Hence, in the final schedule B, we have $\min\{t_1^j, t_2^j\} \leq t^j \leq \max\{t_1^j, t_2^j\}$ for each $1 \leq j \leq m$. This completes the proof of the lemma. $\square$

Using Lemma 12, we now prove a lower bound on the competitive ratio of any IP-clairvoyant algorithm.

Theorem 8 *Any IP-clairvoyant algorithm is $\Omega(\ln^{1-1/\alpha} P)$-competitive with respect to makespan plus energy, where P is the total number of processors.*

Proof. Consider any set $\mathcal{J}$ of P sequential jobs with identical release time and whose total work satisfies $w(J_1) < w(J_2) < \cdots < w(J_P)$. Since the number of jobs is the same as the number of processors, we can assume that any IP-clairvoyant algorithm A assigns exactly one job to each processor. Otherwise, we can always shift a job from a processor with two or more jobs to an idle processor, which will not use any more energy while possibly reducing the makespan. In the rest of the proof, we will show that the WCEP algorithm performs no worse than A for any such job set $\mathcal{J}$ under a certain adversarial strategy. Since Theorem 6 showed a lower bound of $\Omega(\ln^{1-1/\alpha} P)$ for WCEP on a particular instance of $\mathcal{J}$, the same lower bound holds for A as well.

Since the only information an IP-clairvoyant algorithm has about a set of jobs is their instantaneous parallelism, all the jobs are indistinguishable to A. Thus, the adversary is free to choose which processor each job is assigned to. In particular, the adversarial strategy is to always assign job J_i, where $i = 1, 2, \cdots, P$, to the processor that first completes $w(J_i)$ amount of work, with ties broken arbitrarily. For convenience, we assume that job J_i is assigned to the i'th processor. Now, to show $H_{\mathsf{WCEP}}(\mathcal{J}) \leq H_{\mathsf{A}}(\mathcal{J})$ under such an adversary, we transform schedule A to WCEP step by step without increasing the total cost. For each $i = 1, 2, \cdots, P$, we divide job J_i into i segments. The j'th segment has work $w(J_j) - w(J_{j-1})$, for $1 \leq j \leq i$, and $w(J_0)$ is defined to be 0. Let t_i^j denote the completion time of the j'th segment of job J_i in schedule A. By the adversarial strategy, we have $t_i^i \leq t_k^i$ for all $i \leq k \leq P$.

First, we construct schedule A' from A by averaging the execution speed for each segment of each job. By the convexity of the power function, the completion time of all the segments will remain the same in A', but the energy consumption may be reduced. We then get schedule A'' from A' by iteratively performing the following two-step transformation for each $i = P-1, P-2, \cdots, 1$: (1) Slow down the execution of the last segment of job J_i until its completion time t_i^i is equal to t_{i+1}^i. (2) Apply Lemma 12 to get a potentially more energy-efficient schedule that executes the first i segments of all jobs in $\{J_i, \cdots, J_P\}$ identically. Note that, after each iteration i, the corresponding segments of all jobs in $\{J_i, \cdots, J_P\}$ will be completed at the same time, so that the first $i-1$ segments of them can be collectively considered as a single job in the next iteration when applying Lemma 12 in step (2). Also notice that, for each $j = 1, \cdots, i-1$ after iteration i, the completion time t^j for the j'th segment of all jobs in $\{J_i, \cdots, J_P\}$ satisfies $t^j \geq t_j^j$ by Lemma 12 and the adversarial strategy, so that we can apply step (1) in the subsequent iterations. Therefore, at the end of the last iteration, the corresponding segments of all jobs are aligned. Moreover, schedule A'' apparently has the same makespan as A' but may consume less energy. Now, we apply Lemma 8 to construct a schedule B from A'' such that it consumes constant total power $\frac{1}{\alpha-1}$ at any time, and has $H_{\mathsf{B}}(\mathcal{J}) \leq H_{\mathsf{A}''}(\mathcal{J})$. By observing that B is identical to WCEP, the proof is complete. □

5 Discussions and Conclusion

In this paper, we considered energy-efficient scheduling for parallel jobs on multiprocessor systems. We have given state-of-the-art results for the objective of total flow time plus energy in both non-clairvoyant and IP-clairvoyant settings. Moreover, we have studied, for the first time in the literature, makespan plus energy as an objective function. Tight bounds have been proven in this case under the IP-clairvoyant setting.

As mentioned previously, the MultiLaps algorithm proposed by Chan, Edmonds and Pruhs [16] has the same upper and lower bounds as our N-EQUI algorithm with respect to total flow time plus energy. However, their results are based on a different execution model than ours. It would be interesting to further study the relationship between the two models, and to close the gap between the upper and lower bounds for arbitrarily released jobs under either model.

For the objective of makespan plus energy, we have studied the performance of IP-clairvoyant algorithms. The natural question is to consider non-clairvoyant scheduling. Previous studies have shown that, for minimizing makespan alone, a 2-competitive algorithm exists in the IP-clairvoyant setting [13], whereas any non-clairvoyant algorithm is at least

Table 2: Competitive ratios of non-clairvoyant and IP-clairvoyant algorithms with respect to total response time plus energy and makespan plus energy.

	Non-clairvoyant	IP-clairvoyant
Total response time plus energy	$\Omega(\ln^{1/\alpha} P)$	$O(1)$
Makespan plus energy	*Open problem*	$\Omega(\ln^{1-1/\alpha} P)$

$\Omega(\ln n / \ln \ln n)$-competitive [42]. Moreover, by comparing the known competitive ratios of IP-clairvoyant and non-clairvoyant algorithms with respect to both objective functions as shown in Table 2, we conjecture that minimizing makespan plus energy is more difficult than minimizing total flow time plus energy, and hence is likely to incur a larger lower bound than $\Omega(\ln^{1/\alpha} P)$ in the non-clairvoyant setting.

References

[1] K. Agrawal, C. E. Leiserson, Y. He, and W-J. Hsu. Adaptive work-stealing with parallelism feedback. *ACM Transactions on Computer Systems*, 26(3):7:1–7:32, 2008.

[2] S. Albers. Energy-efficient algorithms. *Communications of the ACM*, 53(5):86–96, 2010.

[3] S. Albers and H. Fujiwara. Energy-efficient algorithms for flow time minimization. *ACM Transactions on Algorithms*, 3(4):1–17, 2007.

[4] L. L. Andrew, A. Wierman, and A. Tang. Optimal speed scaling under arbitrary power functions. *ACM SIGMETRICS Performance Evaluation Review*, 37(2):39–41, 2009.

[5] L. L. Andrew, M. Lin, and A. Wierman. Optimality, fairness, and robustness in speed scaling designs. *ACM SIGMETRICS Performance Evaluation Review*, 38(1):37–48, 2010.

[6] N. Bansal, H.-L. Chan, and K. Pruhs. Speed scaling with an arbitrary power function. *ACM Transactions on Algorithms*, 9(2):18, 2013.

[7] N. Bansal, K. Pruhs, and C. Stein. Speed scaling for weighted flow time. *SIAM Journal on Computing*, 39(4):1294–1308, 2009.

[8] L. Becchetti and S. Leonardi and A. Marchetti-Spaccamela and K. Pruhs. Semi-clairvoyant scheduling. *Theoretical Computer Science*, 324(2-3):325–335, 2004.

[9] M. A. Bender and M. O. Rabin. Online scheduling of parallel programs on heterogeneous systems with applications to Cilk. *Theory of Computing Systems*, 35(3):289-304, 2002.

[10] R. D. Blumofe and C. E. Leiserson. Scheduling multithreaded computations by work stealing. *Journal of the ACM*, 46(5):720–748, 1999.

[11] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis.* Cambridge University Press, New York, NY, USA, 1998.

[12] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[13] T. Brecht, X. Deng, and N. Gu. Competitive dynamic multiprocessor allocation for parallel applications. *Parallel Processing Letters*, 7(1):89-100, 1997.

[14] T. Brecht and K. Guha. Using parallel program characteristics in dynamic processor allocation policies. *Performance Evaluation*, 4(27-28):519–539, 1996.

[15] D. M. Brooks, P. Bose, S. E. Schuster, H. Jacobson, P. N. Kudva, A. Buyuktosunoglu, J.-D. Wellman, V. Zyuban, M. Gupta, and P. W. Cook. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *IEEE Micro*, 20(6):26–44, 2000.

[16] H.-L. Chan, J. Edmonds, T.-W. Lam, L.-K. Lee, A. Marchetti-Spaccamela, and K. Pruhs. Nonclairvoyant speed scaling for flow and energy. *Algorithmica*, 61(3): 507-517, 2011.

[17] H.-L. Chan, J. Edmonds, and K. Pruhs. Speed scaling of processes with arbitrary speedup curves on a multiprocessor. *Theory of Computing Systems*, 49(4):817-833, 2011.

[18] S.-H. Chan, T.-W. Lam, and L.-K. Lee. Non-clairvoyant speed scaling for weighted flow time. In *ESA*, pages 23–35, Liverpool, UK, 2010.

[19] H.-L. Chan, T.-W. Lam, and R. Li. Energy-efficient due date scheduling. In *TAPAS*, pages 69–80, Rome, Italy, 2011.

[20] Z.-L. Chen. Simultaneous job scheduling and resource allocation on parallel machines. *Annals of Operations Research*, 129:135–153, 2004.

[21] X. Deng, N. Gu, T. Brecht, and K. Lu. Preemptive scheduling of parallel jobs on multiprocessors. *SIAM Journal on Computing*, 30(1):145–160, 2000.

[22] J. Edmonds. Scheduling in the dark. *Theoretical Computer Science*, 235(1):109–141, 2000.

[23] J. Edmonds, D. D. Chinn, T. Brecht, and X. Deng. Non-clairvoyant multiprocessor scheduling of jobs with changing execution characteristics. *Journal of Scheduling*, 6(3):231–250, 2003.

[24] J. Edmonds and K. Pruhs. Scalably scheduling processes with arbitrary speedup curves. *ACM Transactions on Algorithms*, 8(3):28, 2012.

[25] G. Greiner, T. Nonner, and A. Souza. The bell is ringing in speed-scaled multiprocessor scheduling. In *SPAA*, pages 11–18, Calgary, AB, Canada, 2009.

[26] D. Grunwald, I. Charles B. Morrey, P. Levis, M. Neufeld, and K. I. Farkas. Policies for dynamic clock scheduling. In *OSDI*, pages 6–6, San Diego, CA, USA, 2000.

[27] G. H. Hardy, J. E. Littlewood, and G. Pólya. *Inequalities.* Cambridge University Press, second edition, 1988.

[28] Y. He, W-J. Hsu, C. E. Leiserson. Provably efficient online nonclairvoyant adaptive scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 19(9):1263–1279, 2008.

[29] S. Herbert and D. Marculescu. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *ISLPED*, pages 38–43, Portland, OR, USA, 2007.

[30] S. Im, B. Moseley, and K. Pruhs. A tutorial on amortized local competitiveness in online scheduling. *SIGACT News*, 42(2):83–97, 2011.

[31] S. Irani and K. Pruhs. Algorithmic problems in power management. *SIGACT News*, 36(2):63–76, 2005.

[32] J. M. Jaffe. An analysis of preemptive multiprocessor job scheduling. *Mathematics of Operations Research*, 5(3):415–421, 1980.

[33] B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. *Journal of the ACM*, 47(4):617–643, 2000.

[34] W. Kim, and D. Brooks, and G.-Y. Wei. A fully-integrated 3-level DC/DC converter for nanosecond-scale DVS with fast shunt regulation. In *ISSCC*, pages 268–270, San Francisco, CA, USA, 2011.

[35] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks. System level analysis of fast, per-core DVFS using on-chip switching regulators. In *HPCA*, pages 123–134, Salt Lake City, UT, USA, 2008.

[36] T.-W. Lam, L.-K. Lee, I. K. K. To, and P. W. H. Wong. Nonmigratory multiprocessor scheduling for response time and energy. *IEEE Transactions on Parallel and Distributed Systems*, 19(11):1527–1539, 2008.

[37] T.-W. Lam, L.-K. Lee, I. K.-K. To, and P. W. H. Wong. Online speed scaling based on active job count to minimize flow plus energy. *Algorithmica*, 65(3):605–633, 2013.

[38] T. Mudge. Power: A first-class architecture design constraint. *Computer*, 34(4):52–58, 2001.

[39] K. Pruhs. Competitive online scheduling for server systems. *ACM SIGMETRICS Performance Evaluation Review*, 34(4):52–58, 2007.

[40] K. Pruhs, and C. Stein. How to schedule when you have to buy your energy. In *APPROX-RANDOM*, pages 352-365, Barcelona, Spain, 2010.

[41] K. Pruhs, R. van Stee, and P. Uthaisombut. Speed scaling of tasks with precedence constraints. *Theory of Computing Systems*, 43(1):67–80, 2008.

[42] J. Robert and N. Schabanel. Non-clairvoyant batch set scheduling: Fairness is fair enough. In *ESA*, pages 741–753, Eilat, Israel, 2007.

[43] D. B. Shmoys and Éva Tardos. Scheduling unrelated machines with costs. In *SODA*, pages 448–454, Austin, Texas, USA, 1993.

[44] H. Sun, Y. Cao, and W.-J. Hsu. Non-clairvoyant speed scaling for batched parallel jobs on multiprocessors. In *CF*, pages 99–108, Ischia, Italy, 2009.

[45] H. Sun, Y. Cao, and W.-J. Hsu. Efficient adaptive scheduling of multiprocessors with stable parallelism feedback. *IEEE Transactions on Parallel and Distributed Systems*, 22(4):594–607, 2011.

[46] H. Sun, Y. He, and W.-J. Hsu. Speed scaling for energy and performance with instantaneous parallelism. In *TAPAS*, pages 240–251, Rome, Italy, 2011.

[47] M. A. Trick. Scheduling multiple variable-speed machines. In *IPCO*, pages 485–494, Waterloo, Canada, 1990.

[48] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced CPU energy. In *OSDI*, pages 13–23, Monterey, CA, USA, 1994.

[49] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *FOCS*, pages 374–382, Milwaukee, WI, USA, 1995.

[50] X. Zhang, K. Shen, S. Dwarkadas, and R. Zhong. An evaluation of per-chip nonuniform frequency scaling on multicores. In *USENIXATC*, pages 19–19, Boston, MA, USA, 2010.

[51] X. Zhao and N. Jamali. Fine-grained per-core frequency scheduling for power efficient-multicore execution. In *IGCC*, pages 1–8, Orlando, FL, USA, 2011.