1 Definitions

An instance consists of \( n \) jobs, where job \( i \) has a release time \( r_i \). Implicitly each job has a work/size of 1 and a weight of 1. We assume, without loss of generality, that \( r_1 \leq r_2 \leq \ldots \leq r_n \). An online scheduler is not aware of job \( i \) until time \( r_i \), and, at time \( r_i \), learns \( y_i \). For each time, a schedule specifies a job to be run and a speed at which the job is run. A job \( i \) completes once 1 unit of work has been performed on \( i \). The speed is the rate at which work is completed; a job with work \( y \) run at a constant speed \( s \) completes in \( \frac{y}{s} \) seconds. Every nonnegative real number is an allowable speed. The power consumed when running at speed \( s \) is \( s^2 \).

The energy used is power integrated over time. We assume that preemption is allowed, that is, a job may be suspended and later restarted from the point of suspension. A job is active at time \( t \) if it has been released but not completed at time \( t \).

Let \( X \) be an arbitrary algorithm. Let \( w_x(t) \) denote the aggregate fractional weight of the active jobs at time \( t \) for algorithm \( X \). So for example, if a job \( i \) is \( 2/3 \) completed at time \( t \), then its fractional weight at time \( t \) is \( 1/3 \). Let \( s_x(t) \) be the speed at time \( t \) for algorithm \( x \), and let \( p_x(t) = (s_x(t))^2 \) be the power consumed at time \( t \) by algorithm \( X \). Let \( E_x(t) = \int_{k=0}^{t} p_x(k)\,dk \) be the energy spent up until time \( t \) by algorithm \( A \).

Let \( W_x(t) = \int_{k=0}^{t} w_x(k)\,dk \) be the fractional weighted flow up until time \( t \) for algorithm \( X \). Our objective function combines flow and energy and we let \( G_x(t) = W_x(t) + E_x(t) \) be the fractional weighted flow and energy up until time \( t \) for algorithm \( X \). Let \( E_x = E_x(\infty) \), \( W_x = W_x(\infty) \), \( G_x = G_x(\infty) \) be the energy, fractional weighted flow, and fractional weighted flow plus energy, respectively, for algorithm \( X \). We use \( \text{Opt} \) to denote the offline adversary, and subscript a variable by “\( o \)” to denote the value of a variable for the adversary. So \( W_o \) is the fractional weighted flow for the adversary.

2 Amortized Local Competitiveness

A common notion to measure an on-line scheduling algorithm is local competitiveness, meaning roughly that the algorithm is competitive at all times during the execution. Local competitiveness is generally not achievable in speed scaling problems because the adversary may spend essentially all of its energy in some small period of time, making it impossible for any online algorithm to be locally competitive at that time. Thus, we will analyze our algorithms using amortized local competitiveness, which we now define. Let \( X \) be an arbitrary on-line scheduling algorithm, and \( H \) an arbitrary objective function. Let \( \frac{dH(t)}{dt} \) be the rate of increase of the objective \( H \) at time \( t \). The online algorithm \( X \) is amortized locally \( \gamma \)-competitive with potential function \( \Phi(t) \) for objective function \( H \) if the following two conditions hold:

**Boundary Condition:** \( \Phi \) is initially 0, and and finally nonnegative. That is, \( \Phi(0) = 0 \), and there exists some time \( t' \) such that for all \( t \geq t' \) it is the case that \( \Phi(t) \geq 0 \).

**General Condition:** For all times \( t \),

\[
\frac{dH_x(t)}{dt} - \gamma \frac{dH_o(t)}{dt} + \frac{d\Phi(t)}{dt} \leq 0.
\]

(1)

We break the general condition into three cases:

**Running Condition:** For all times \( t \) when no job arrives (1) holds.

**Job Arrival Condition:** \( \Phi \) does not increase when a new job arrives.
Completion Condition: $\Phi$ does not increase when either the online algorithm or the adversary complete a job.

Observe that when $\Phi(t)$ is identically zero, we have ordinary local competitiveness. It is well known that amortized local $\gamma$-competitiveness implies that when the algorithm completes, the total cost of the online algorithm is at most $\gamma$ times the total cost of the optimal offline algorithm.

Lemma 1 If online algorithm $X$ is amortized locally $\gamma$-competitive with potential function $\Phi(t)$ for objective function $H$, then $H_x \leq \gamma H_o$.

Proof: Let $t_1, \ldots, t_{3n}$ be the events that either a job is released, the online algorithm $x$ completes a job, or the adversary completes a job. Let $\Delta(\Phi(t_i))$ denote the change in potential in response to event $t_i$. Let $t_0 = 0$ and $r_{3n+1} = +\infty$. Integrating equation 1 over time, we get that

$$H_x + \sum_{i=1}^{3n+1} \Delta(\Phi(t_i)) \leq \gamma H_o.$$ 

By the job arrival condition, and the completion condition, we can conclude that $H_x + \Phi(\infty) - \Phi(0) \leq \gamma H_o$, and finally, by the boundary condition, we can conclude that $H_x \leq \gamma H_o$.

Now consider the case that the objective function is $G$, the fractional weighted flow plus energy. Then $\frac{dG(t)}{dt} = w(t) + p(t) = w(t) + s(t)^2$, and equation 1 is equivalent to:

$$w_x(t) + s_x(t)^2 - \gamma(w_o(t) + s_o(t)^2) + \frac{d\Phi(t)}{dt} \leq 0.$$ 

(2)

For our purposes, we will always consider the algorithm $A$, where $s_a(t)^2 = w_a(t)$. Thus equation 2 is equivalent to:

$$2w_a(t) - \gamma(w_o(t) + s_o(t)^2) + \frac{d\Phi(t)}{dt} \leq 0.$$ 

(3)

If $w_o(t) + s_o(t)^2 = 0$ then equation 3 is equivalent to

$$\frac{d\Phi(t)}{dt} \leq -2w_a.$$ 

(4)

Thus, we are essentially required to pick a potential function satisfying equation 4. Note that if $w_a(t) + s_o(t)^2 = 0$ then it must be the case that the adversary has no active jobs at time $t$, and $w_o(t) = s_o(t)^2 = 0$. If $w_o(t) + s_o(t)^2 \neq 0$ then equation 3 can be rewritten as

$$\gamma \geq \frac{2w_a(t) + \Phi(t)}{w_o(t) + s_o(t)^2}.$$ 

(5)

Since we want to choose $\gamma$ to be as small as possible, while still satisfying inequality 5, the right side of this inequality will denote our competitive ratio.
3 The Result

We first show that the speed scaling algorithm $A$, where $s_a(t) = w_a(t)^{1/2}$ is 2-competitive for the objective function of fractional flow time plus energy.

We first recall the following classic inequality and its corollary.

**Theorem 2 (Young’s Inequality)** Let $f$ be a real-valued, continuous, and strictly increasing function on $[0, c]$ with $c > 0$. If $f(0) = 0$, and $a, b$ such that $a \in [0, c]$, and $b \in [0, f(c)]$, then

$$\int_0^a f(x)dx + \int_0^b f^{(-1)}(x)dx \geq ab,$$

where $f^{(-1)}$ is the inverse function of $f$.

**Corollary 3** For positive reals $a, b, \mu, p$ and $q$ such that $1/p + 1/q = 1$, the following holds:

$$\mu \frac{a^p}{p} + \left(\frac{1}{\mu}\right)^{q/p} \frac{b^q}{q} \geq ab.$$

Note that for $\mu = 1$, this is the classic Hölder’s inequality.

We now prove the main result of this section.

**Theorem 4** The speed scaling algorithm $A$, where $s_a(t) = w_a(t)^{1/\alpha}$ is 2-competitive with respect to the objective $G$ of fractional flow plus energy.

**Proof:** We prove that algorithm $A$ is amortized locally 2-competitive using the potential function

$$\Phi(t) = \frac{8}{3} (\max(0, w_a(t) - w_o(t)))^{3/2}.$$

We first need to verify the boundary condition. Clearly $\Phi(0) = 0$, as $w_a(0) = w_o(0)$, and $\Phi(t)$ is always non-negative. $\Phi$ satisfies the job completion condition since the fractional weight of a job approaches zero continuously as the job nears completion and there is no discontinuity in $w_a(t)$ or $w_o(t)$ when a job completes. $\Phi$ satisfies the job arrival condition since both $w_a(t)$ and $w_o(t)$ increase simultaneously by 1 when a new job arrives.

We are left to establish the running condition. We now break the argument into two cases. In the first case assume that that $w_a(t) < w_o(t)$. This case is simpler, since the offline adversary has large fractional weight. Here $\Phi(t) = 0$ and $\frac{d\Phi(t)}{dt} = 0$ by the definition of $\Phi$. Since we know that $w_a(t) < w_o(t)$, it must be the case that $w_o(t) + s_a(t)^2 \neq 0$, and then that the right side of equation 5 is clearly at most 2.

We now turn to the interesting case that $w_a(t) \geq w_o(t)$. For notational ease, we will drop the time $t$ from the notation, since all variables are understood to be functions of $t$. We consider $d\Phi/dt$.

$$\frac{d\Phi}{dt} = \frac{8}{3} \frac{d}{dt} \left((w_a - w_o)^{3/2}\right) = 4(w_a - w_o)^{1/2} \frac{d(w_a - w_o)}{dt}.$$

Since jobs have unit density, the rate at which the fractional weight decreases is exactly the rate at which unfinished work decreases, which is just the speed of the algorithm. Thus $\frac{dw}{dt} = -s$. Moreover since $s_a = w_a^{1/2}$, by the definition of $A$, equation 6 can be written as

$$\frac{d\Phi}{dt} = -4(w_a - w_o)^{1/2}(s_a - s_o) = -4(w_a - w_o)^{1/2}(w_a^{1/2} - s_o).$$

3
Since \( w_a \geq w_a - w_o \), it follows that \(-w_a^{1/2} \leq -(w_a - w_o)^{1/2}\) and equation 7 implies that

\[
\frac{d\Phi}{dt} \leq -4(w_a - w_o) + 4(w_a - w_o)^{1/2}s_o
\]  

(8)

Applying Young’s inequality (cf. Corollary 3) with \( \mu = 1, a = s_0, p = 2, b = (w_a - w_o)^{1/2} \), and \( q = \frac{1}{2} \), we obtain that \((w_a - w_o)^{1/2}s_o \leq \frac{1}{2}(w_a - w_o) + s_o^2/2\). Thus equation 8 can be written as

\[
\frac{d\Phi}{dt} \leq -4(w_a - w_o) + 2(w_a - w_o) + 2s_o^2 = -2(w_a - w_o) + 2s_o^2
\]  

(9)

If \( w_o + s_o^2 = 0 \) then equation 9 implies that \( \frac{d\Phi}{dt} \leq -2w_a \), and equation 4 holds. If \( w_o + s_o^2 \neq 0 \) then, plugging equation 9 into the right side of equation 5, we obtain a bound on the competitive ratio of

\[
\frac{2w_a + \frac{d\Phi}{dt}}{w_o + s_o^2} \leq \frac{2w_o + (-2w_a + 2w_o + 2s_o^2)}{w_o + s_o^2} = \frac{2w_o + 2s_o^2}{w_o + s_o^2} = 2
\]  

(10)