A Competitive Algorithm for Minimizing Weighted Flow Time on Unrelated Machines with Speed Augmentation

Jivitej S. Chadha and Naveen Garg* and Amit Kumar† and V. N. Muralidhara
Computer Science and Engineering
Indian Institute of Technology Delhi
New Delhi – 110016, India
{cs5040213,naveen,amitk,murali}@cse.iitd.ac.in

ABSTRACT

We consider the online problem of scheduling jobs on unrelated machines so as to minimize the total weighted flow time. This problem has an unbounded competitive ratio even for very restricted settings. In this paper we show that if we allow the machines of the online algorithm to have ϵ more speed than those of the offline algorithm then we can get an $O((1+\epsilon^{-1})^2)$-competitive algorithm.

Our algorithm schedules jobs preemptively but without migration. However, we compare our solution to an offline algorithm which allows migration. Our analysis uses a potential function argument which can also be extended to give a simpler and better proof of the randomized immediate dispatch algorithm of Chekuri-Goel-Khanna-Kumar for minimizing average flow time on parallel machines.

Categories and Subject Descriptors

F.2.2 [**ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY**]: Nonnumerical Algorithms and Problems—*Sequencing and scheduling*

General Terms

Algorithms, Performance

Keywords

Scheduling, flow-time, competitive ratio, approximation algorithms

1. INTRODUCTION

The problem of online scheduling of jobs on multiple machines has been well studied both in the online and scheduling communities. A natural measure to consider in this setting is the weighted sum of flow times of the jobs, where the flow time of a job is defined as the difference in the completion time and the arrival time (also known as release time) of the job. However, most of the research in this area has addressed only the setting of identical multiple machines.

Garg and Kumar [11] were the first to consider the problem for related machines. This is the setting when machine i has speed s_i so that it finishes one unit of processing in time $1/s_i$. Garg and Kumar [10] gave an $O(\log^2 P)$-competitive algorithm for this problem which has recently been improved to $O(\log P)$ [1].

For the more general setting of unrelated machines when the time required to process a job j on machine i equals p_{ij} (and this could be an arbitrary quantity), [12] showed that no competitive algorithm is possible. Their example which showed that no online algorithm can be competitive used only 3 machines and jobs that had unit processing time on 2 of the 3 machines.

In light of this strong negative result we ask ourselves if it is possible to obtain a bounded competitive ratio by providing the online algorithm with slightly more resources than those of the offline algorithm. In particular, we assume that each machine of the online algorithm has ϵ more speed which implies that the work that the offline algorithm can do in 1 unit of time, will require $(1+\epsilon)^{-1}$ time for the online algorithm. In this setting we are able to show a simple online algorithm that is $O((1+\epsilon^{-1})^2)$-competitive.

Our paper also highlights the immense additional power that a small augmentation of speed can provide to the online algorithm. In particular, we get the same competitive ratio for weighted flow time (every job has a weight which may depend on the machine on which the job is scheduled). The schedule that our algorithm provides is non-migratory – a job once it is scheduled on a machine cannot migrate to another machine. However, the offline adversary can be stronger, it can migrate jobs from one machine to another. This shows that speed can also offset the benefits that might accrue with migration. One another nice property of our algorithm is that it dispatches a job to the appropriate machine as soon as the job arrives. We would like to point out that in the absence of speed augmentation we do not even know of any approximation algorithm (with non-trivial guarantees) for the problem of minimizing flow time on unrelated machines [13].

The key technical contribution of this paper is a simple potential function argument that we believe can be applied to other problems as well. In particular, we provide a simple

*Work partially supported by the Max-Planck partner group on "Approximation Algorithms"

†Work partially supported by the Max-Planck partner group on "Approximation Algorithms"

STOC'09, May 31–June 2, 2009, Bethesda, Maryland, USA.

proof of the randomized immediate dispatch algorithm of Chekuri et.al. [9] for minimizing average weighted flow time on parallel machines which also improves the competitive ratio from $O(\epsilon^{-3})$ to $O(\epsilon^{-2})$.

Related Work Speed augmentation in online scheduling problems was first considered by Kalyanasundaram and Pruhs [14] who used it to get improved on-line algorithms for minimizing average (unweighted) flow-time on a single machine in the non-clairvoyant setting. They obtained an $O(\epsilon^{-1})$-competitive algorithm for this problem. Without resource augmentation no online algorithm for this problem can be $\Omega(\log n)$-competitive [17]. Bechetti and Leonardi [7] showed that the randomized multi-level-feedback algorithm is $O(\log n)$ competitive for single machines and $O(\log n \log P)$-competitive for parallel machines in the non-clairvoyant setting. Bansal and Pruhs [5] showed that in the clairvoyant setting some well-known algorithms like SRPT and Shortest Job First are ϵ^{-1}-competitive for the L_p norm of flow-time for all values of $p \geq 1$.

There has been considerable work on minimizing average (unweighted) flow-time on parallel machines [16, 3, 4, 2] – these results established $O(\min(\log P, \log n/m))$-competitive algorithms for this problem. Leonardi and Raz [16] also showed that no randomized on-line algorithm can do better. With ε-resource augmentation, Chekuri et.al. [9] showed that the immediate dispatch non-migratory algorithm of Azar et.al. [2] is $O(\epsilon^{-1})$-competitive for all L_p norms. They also showed that the following simple stateless algorithm has competitive ratio $O(\frac{\log \frac{1}{\varepsilon}}{\varepsilon^3})$ for the average (unweighted) flow-time : when a job arrives, assign it to a machine picked uniformly at random. As mentioned above, Garg and Kumar showed that there cannot be a bounded competitive ratio if we make the model slightly more by adding the restriction that each job can go only on a subset of machines.

For the problem of minimizing weighted flow time on parallel machines, Chekuri, Khanna and Zhu [8] showed that no online algorithm can have a bounded competitive ratio. Ours is the first non-trivial competitive ratio for minimizing weighted flow-time when there are more than one machine.

More recently, Bansal et. al. [6] show that speed augmentation can help us overcome the hardness imposed by non-preemption [15]; they show a polynomial time $O(1)$-approximation algorithm for minimizing flow time on a single machine with $O(1)$ additional speed when no preemption is allowed.

2. PRELIMINARIES AND THE SCHEDULING ALGORITHM

We consider the on-line problem of minimizing total weighted flow-time on unrelated machines. Job j is released at time r_j and requires p_{ij} units of processing if scheduled on machine i. Further, job j has a weight w_{ij} if it is scheduled on machine i. Let $d_{ij} = w_{ij}/p_{ij}$ denote the density of job j on machine i. We shall use $\mathcal{A}$ to denote the schedule produced by our algorithm, and $\mathcal{O}$ the optimal schedule OPT. We assume that a machine in our algorithm has $(1+\varepsilon')$ more speed than the corresponding machine in OPT. Let ε be such that $(1+\varepsilon)^2 = 1+\varepsilon'$.

2.1 Fractional Flow time

In the following sections, we shall compare the fractional flow-time of our algorithm to that of the optimal schedule. Let $x_{i,j,t}$ be a variable between 0 and 1 which tells us the extent to which job j is scheduled on machine i in the period $[t, t+1)$. $x_{i,j,t} = 1$ denotes that all of the t^{th} time-slot on machine i is occupied by job j. The (weighted) fractional flow-time of job j is then defined as

$$F_j = \sum_{i,t} w_{ij} \cdot x_{i,j,t} \left(\frac{1}{2} + \frac{t - r_j}{p_{ij}} \right)$$

The fractional flow time of a schedule $\mathcal{A}$ is denoted by $F^{\mathcal{A}}$ and equals the sum of the fractional flow times of the jobs. The following three claims follow as an easy consequence of the definition of fractional flow time.

CLAIM 2.1. *In any schedule the fractional flow time of a job would be at most its (integral) flow time.*

CLAIM 2.2. *To minimize fractional flow time, each machine should process the jobs assigned to it in decreasing order of density.*

In our schedule, $\mathcal{A}$, job j would be assigned entirely to one machine, say i. In this scenario F_j, as defined above, is equivalent to

$$\frac{w_{ij} p_{ij}}{2} + \int_{r_j}^{\infty} d_{ij} p_{ij}^{\mathcal{A}}(t)\, dt \tag{1}$$

where $p_{ij}^{\mathcal{A}}(t)$ is the remaining processing time of job j on machine i at time t in schedule $\mathcal{A}$. In fact, in the above expression the first term is at most the second term [10]. Let $P^{\mathcal{A}} = \sum_j w_{ij} p_{ij}$ denote the total weighted processing time of schedule $\mathcal{A}$. Hence, $F^{\mathcal{A}} \geq P^{\mathcal{A}}$.

2.2 Relation between fractional and integral flow time

Let $\mathcal{B}$ be a non-migratory schedule. Consider a job j scheduled on machine i in this schedule. Let β_j be the time at which a $(1+\varepsilon)^{-1}$ fraction of job j is completed. Then the fractional flow time of j is at least $\varepsilon w_{ij}(\beta_j - r_j)/(1+\varepsilon)$.

Suppose we increase the speed of each machine by a factor $(1+\varepsilon)$. We modify the schedule $\mathcal{B}$ to get a schedule $\mathcal{B}'$ for these faster machines as follows. Job j is scheduled in the first $p_{ij}/(1+\varepsilon)$ slots of the p_{ij} slots in which this job is scheduled in $\mathcal{B}$. Hence the completion time of j in $\mathcal{B}'$ is β_j and the flow time is $w_{ij}(\beta_j - r_j)$. This implies that for each job, its flow time in $\mathcal{B}'$ is less than $(1+\varepsilon^{-1})$ times its fractional flow time in $\mathcal{B}$.

Henceforth, we would only be interested in the fractional flow time of $\mathcal{A}$.

2.3 The Algorithm

Each machine maintains a queue of jobs assigned to it. At time t let $\mathcal{A}_i(t)$ be the set of unfinished jobs scheduled on machine i in our algorithm. We can think of OPT also as maintaining a queue of jobs for each machine. When a job arrives at time t, it goes to the queue of the machine on which it eventually gets processed. Let $\mathcal{O}_i(t)$ be the set of unfinished jobs scheduled on machine i in OPT.

Let $p_j^{\mathcal{A}}(t)$ be the residual processing time of job j at time t in our algorithm and $p_j^{\mathcal{O}}(t)$ be the corresponding quantity in OPT.

Suppose a job k arrives at time t. For each machine i, we compute the quantity

$$Q(i,k) = w_{ik} \sum_{\substack{j \in A_i(t) \\ d_{ij} > d_{ik}}} p_j^{\mathcal{A}}(t) + p_{ik} \sum_{\substack{j \in A_i(t) \\ d_{ij} \leq d_{ik}}} d_{ij} p_j^{\mathcal{A}}(t)$$

Job k is assigned to the machine i for which $Q(i,k)$ is minimum. This specifies the rule for assigning a job to a machine. Since each machine processes jobs assigned to it in decreasing order of density, the quantity $Q(i,k)$ measures the increase in the flow-time of jobs which are in the queue of machine i (at time t) if we assign k to machine i. Note that there is no increase in the flow-time of jobs of density more than d_{ik}, but the flow time of jobs whose density is less than or equal to d_{ik} increases by p_{ik} times their weight.

3. ANALYSIS

Our analysis relies on a potential function argument. Define

$$\begin{aligned}
&\Phi_i(t) \\
= &\sum_{k \in A_i(t)} p_k^{\mathcal{A}}(t) \left(\sum_{\substack{j \in A_i(t) \\ d_{ij} \leq d_{ik}}} d_{ij} p_j^{\mathcal{A}}(t) - \sum_{\substack{j \in O_i(t) \\ d_{ij} \leq d_{ik}}} d_{ij} p_j^{\mathcal{O}}(t) \right) \\
- &\sum_{k \in O_i(t)} p_k^{\mathcal{O}}(t) \left(\sum_{\substack{j \in A_i(t) \\ d_{ij} \leq d_{ik}}} d_{ij} p_j^{\mathcal{A}}(t) - \sum_{\substack{j \in O_i(t) \\ d_{ij} \leq d_{ik}}} d_{ij} p_j^{\mathcal{O}}(t) \right)
\end{aligned}$$

and let $\Phi = \sum_i \Phi_i$. Φ is zero both at the start and the end of the algorithm. $\Phi(t)$ is continuous at all times other than at the arrival times of jobs. $\Phi(t)$ is also differentiable almost-everywhere (the slope changes only when a job finishes).

We show that the total decrease in Φ is at least $\varepsilon(F^{\mathcal{A}} - P^{\mathcal{A}}/2 - F^{\texttt{OPT}})$ (Lemma 3.1). We also show that the total increase in Φ is at most $F^{\texttt{OPT}}$ (Lemma 3.2). Since total decrease in Φ equals the total increase in ϕ, we have,

$$\begin{aligned}
\varepsilon(F^{\mathcal{A}} - F^{\texttt{OPT}} - P^{\mathcal{A}}/2) &\leq \text{Decrease in } \Phi \\
&= \text{Increase in } \Phi \\
&\leq F^{\texttt{OPT}}
\end{aligned}$$

Since $F^{\mathcal{A}} \geq P^{\mathcal{A}}$, this implies

$$F^{\mathcal{A}} \leq 2(1+\varepsilon^{-1}) F^{\texttt{OPT}}$$

We now prove the two lemmas. Let $t_1 < t_2 < \cdots < t_\ell$ be the times at which jobs are released.

LEMMA 3.1. *The sum of the decreases in Φ in the open intervals $(t_l, t_{l+1}), \forall l$ is at least $\varepsilon(F^{\mathcal{A}} - F^{\texttt{OPT}})$.*

PROOF. From time t to $t+\delta$ we calculate the decrease in Φ_i. Let our algorithm schedule job a on machine i and OPT schedule job b. Note that a is the highest density job in $A_i(t)$ while b would be the highest density job in $O_i(t)$. We consider 2 cases depending on which of a, b has the higher density.

We first consider the case when a has higher density. The residual processing time of a decreases by $\delta(1+\varepsilon)$ and the decrease in Φ_i due to this equals

$$\begin{aligned}
&\delta(1+\epsilon) \left(\sum_{j \in A_i(t)} d_{ij} p_j^{\mathcal{A}}(t) - \sum_{j \in O_i(t)} d_{ij} p_j^{\mathcal{O}}(t) \right) \\
+ \quad &\delta(1+\varepsilon) d_{ia} \left(p_a^{\mathcal{A}}(t) - \delta(1+\varepsilon) \right)
\end{aligned}$$

We assume δ small enough so that $\delta(1+\varepsilon) \leq p_a^{\mathcal{A}}(t)$ and hence the increase in Φ_i is at least the first term. Similarly, the residual processing time of b decreases by δ and the decrease in Φ_i due to this equals

$$\begin{aligned}
&\delta \left(\sum_{j \in O_i(t)} d_{ij} p_j^{\mathcal{O}}(t) - \sum_{\substack{j \in A_i(t) \\ d_{ij} \leq d_{ib}}} d_{ij} p_j^{\mathcal{A}}(t) - \sum_{\substack{j \in A_i(t) \\ d_{ij} > d_{ib}}} d_{ib} p_j^{\mathcal{A}}(t) \right) \\
&\qquad + \delta d_{ib} \left(p_b^{\mathcal{O}}(t) - \delta \right)
\end{aligned}$$

which is greater than

$$\delta \left(\sum_{j \in O_i(t)} d_{ij} p_j^{\mathcal{O}}(t) - \sum_{j \in A_i(t)} d_{ij} p_j^{\mathcal{A}}(t) \right)$$

assuming $\delta \leq p_b^{\mathcal{O}}(t)$.

There is an additional decrease in Φ_i due to the fact that the residual processing times of a and b decrease simultaneously. This additional decrease is in the term $-p_a^{\mathcal{A}}(t) \cdot p_b^{\mathcal{O}}(t) \cdot d_{ib}$ and equals $(1+\varepsilon)\delta^2 d_{ib}$. Therefore the total decrease in Φ_i is at least

$$\delta\varepsilon \left(\sum_{j \in A_i(t)} d_{ij} p_j^{\mathcal{A}}(t) - \sum_{j \in O_i(t)} d_{ij} p_j^{\mathcal{O}}(t) \right)$$

We next consider the case when b has higher density. The decrease in Φ_i due to the decrease in the residual process time of a is now, assuming $\delta(1+\varepsilon) \leq p_a^{\mathcal{A}}(t)$, at least

$$\delta(1+\epsilon) \left(\sum_{j \in A_i(t)} d_{ij} p_j^{\mathcal{A}}(t) - \sum_{\substack{j \in O_i(t) \\ d_{ij} \leq d_{ia}}} d_{ij} p_j^{\mathcal{O}}(t) - \sum_{\substack{j \in O_i(t) \\ d_{ij} > d_{ia}}} d_{ia} p_j^{\mathcal{O}}(t) \right)$$

which is greater than

$$\delta(1+\epsilon) \left(\sum_{j \in A_i(t)} d_{ij} p_j^{\mathcal{A}}(t) - \sum_{j \in O_i(t)} d_{ij} p_j^{\mathcal{O}}(t) \right)$$

Similarly, the residual processing time of b decreases by δ and the decrease in Φ_i due to this, again assuming $\delta \leq p_b^{\mathcal{O}}(t)$, is at least

$$\delta \left(\sum_{j \in O_i(t)} d_{ij} p_j^{\mathcal{O}}(t) - \sum_{j \in A_i(t)} d_{ij} p_j^{\mathcal{A}}(t) \right).$$

Hence the total decrease in Φ_i is, once again, at least

$$\delta\varepsilon \left(\sum_{j \in A_i(t)} d_{ij} p_j^{\mathcal{A}}(t) - \sum_{j \in O_i(t)} d_{ij} p_j^{\mathcal{O}}(t) \right).$$

This implies that the total decrease in Φ during (t_l, t_{l+1})

is at least

$$\int_{t_l}^{t_{l+1}} \varepsilon \sum_i \left(\sum_{j \in A_i(t)} d_{ij} p_j^{\mathcal{A}}(t) - \sum_{j \in O_i(t)} d_{ij} p_j^{\mathcal{O}}(t) \right) dt$$

. Adding over all l, and using (1), we get that the total decrease in Φ during these intervals is at least $\varepsilon(F^{\mathcal{A}} - P^{\mathcal{A}}/2 - F^{\texttt{OPT}})$. □

The potential function increases when jobs arrive.

LEMMA 3.2. *The total increase in Φ at times $t_l, \forall l$ is at most the fractional flow time of* OPT.

PROOF. Let k be a job which arrives at time t and is scheduled by $\mathcal{A}$ on machine 1 and by OPT on machine 2. The increase in Φ_1 equals

$$w_{1k} \sum_{\substack{j \in A_1(t) \\ d_{1j} > d_{1k}}} p_j^{\mathcal{A}}(t) + p_{1k} \sum_{\substack{j \in A_1(t) \\ d_{1j} \le d_{1k}}} d_{1j} p_j^{\mathcal{A}}(t) \quad (2)$$

$$-w_{1k} \sum_{\substack{j \in O_1(t) \\ d_{1j} > d_{1k}}} p_j^{\mathcal{O}}(t) - p_{1k} \sum_{\substack{j \in O_1(t) \\ d_{1j} \le d_{1k}}} d_{1j} p_j^{\mathcal{O}}(t) \quad (3)$$

Similarly increase in Φ_2 is

$$-w_{2k} \sum_{\substack{j \in A_2(t) \\ d_{2j} > d_{2k}}} p_j^{\mathcal{A}}(t) - p_{2k} \sum_{\substack{j \in A_2(t) \\ d_{2j} \le d_{2k}}} d_{2j} p_j^{\mathcal{A}}(t) \quad (4)$$

$$+w_{2k} \sum_{\substack{j \in O_2(t) \\ d_{2j} > d_{2k}}} p_j^{\mathcal{O}}(t) + p_{2k} \sum_{\substack{j \in O_2(t) \\ d_{2j} \le d_{2k}}} d_{2j} p_j^{\mathcal{O}}(t) \quad (5)$$

Note that the quantity in expression (2) equals $Q(1,k)$ while expression (4) equals $-Q(2,k)$. Our choice of machine to schedule job k implies that (2) + (4) ≤ 0. Since (3) < 0, the total increase in Φ is at most (5). But (5) is also the increase in the fractional flow time of OPT due to the arrival of job k. Hence total increase in Φ is at most the fractional flow time of OPT. □

4. COMPARISON WITH MIGRATORY OPTIMUM

So far we have compared our algorithm with a non-migratory optimum. We now show that in fact our algorithms are also competitive with respect to a migratory optimum, mOPT. Let $p_{ij}^{\mathcal{O}}(t)$ denote the residual processing time of job j on machine i at time t in OPT.

We need to re-analyse the increase in Φ when a job arrives. Let k be the job which arrives at time t and is scheduled on machine 1 by $\mathcal{A}$, while mOPT sends α_i ($\sum_i \alpha_i = 1$) fraction of k to machine i. The increase in Φ_1 due to job k being scheduled on machine 1 by $\mathcal{A}$ is as before and equals

$$w_{1k} \sum_{\substack{j \in A_1(t) \\ d_{1j} > d_{1k}}} p_j^{\mathcal{A}}(t) + p_{1k} \sum_{\substack{j \in A_1(t) \\ d_{1j} \le d_{1k}}} d_{1j} p_j^{\mathcal{A}}(t) \quad (6)$$

$$-w_{1k} \sum_{d_{1j} > d_{1k}} p_{1j}^{\mathcal{O}}(t) - p_{1k} \sum_{d_{1j} \le d_{1k}} d_{1j} p_{1j}^{\mathcal{O}}(t) \quad (7)$$

The increase in $\Phi = \sum_i \Phi_i$ due to α_i fraction of k being scheduled on machine i in mOPT equals

$$-\sum_i \alpha_i \left(w_{ik} \sum_{\substack{j \in A_i(t) \\ d_{ij} > d_{ik}}} p_j^{\mathcal{A}}(t) + p_{ik} \sum_{\substack{j \in A_i(t) \\ d_{ij} \le d_{ik}}} d_{ij} p_j^{\mathcal{A}}(t) \right) \quad (8)$$

$$+\sum_i \alpha_i \left(w_{ik} \sum_{d_{ij} > d_{ik}} p_{ij}^{\mathcal{O}}(t) + p_{ik} \sum_{d_{ij} \le d_{ik}} d_{ij} p_{ij}^{\mathcal{O}}(t) \right) \quad (9)$$

Note that we are ignoring the change in Φ_1 due to the fact that job k is assigned on machine 1 by our algorithm and to an extent α_1 by OPT. Accounting for this would only decrease Φ_1.

Our choice of machine to schedule job k implies that (6) + (8) ≤ 0. Since (7) < 0, the total increase in Φ is at most (9). But (9) is also the increase in fractional flow time of mOPT due to the arrival of job k. Hence total increase in Φ is at most the fractional flow time of mOPT. The rest of the analysis continues as before and hence

$$F^{\mathcal{A}} \le 2(1 + \varepsilon^{-1}) F^{\texttt{mOPT}}$$

5. A SIMPLER (AND BETTER) ANALYSIS OF THE RANDOMIZED ALLOCATION ALGORITHM FOR PARALLEL MACHINES

In this section, we consider the setting of parallel identical machines. Consider the following simple algorithm : when a job arrives assign it randomly to one of the machines. For a particular machine, we follow the highest-density first rule to schedule the jobs assigned to this machine.

We use the same definitions as in Section 4. Note that $\mathcal{A}_i(t), p_j^{\mathcal{A}}(t), \Phi_i t$ and $\Phi(t)$ are random variables. We first observe that it is easy to characterize OPT if we allow it to migrate jobs and process a job simultaneously on multiple machines (this can only reduce its flow-time).

LEMMA 5.1. *There is an optimal solution which distributes each job evenly on all the machines.*

PROOF. We can think of the m machines as a single machine with m times the speed. View the optimal solution as a schedule on this single machine. We can get back an optimal solution for the m machine case by splitting each unit time interval in this schedule into m equal parts. □

Suppose job k arrives at time t. We would like to compute the increase in $E[\Phi]$. Fix all the random choices before job k. So $\Phi(t)$ is now a known quantity. Suppose we send k to machine 1 (this happens with probability $1/m$, where m is the number of machines. Then the increase in Φ_1 is at most $Q(1,k)$. Hence, the expected increase in Φ due to scheduling of job k by our algorithm is $(1/m)\sum_i Q(i,k)$.

On the other hand, the increase in Φ due to OPT scheduling $1/m$ fraction of job k on each machine equals (8)+(9) with $\alpha_1 = \alpha_2 = \cdots = \alpha_m = 1/m$. This in turn is equal to $-(1/m)\sum_i Q(i,k)$ plus the increase in the fractional flow time of OPT due to arrival of job k.

Therefore, the expected increase in Φ due to the arrival of job k is at most the increase in the fractional flow time of OPT. Since this is true for all random choices made prior to job k, we can remove the conditioning on the random choices made before job k. Hence, we have that the increase in $E[\Phi]$ is at most the increase in flow time of OPT.

Since,

$$\varepsilon(F^{\mathcal{A}} - F^{\texttt{OPT}} - P^{\mathcal{A}}/2) \leq \text{Decrease in } \Phi$$

we have that

$$\varepsilon(E[F^{\mathcal{A}} - P^{\mathcal{A}}/2] - F^{\texttt{OPT}}) \leq \text{Decrease in } E[\Phi]$$

The rest of the argument proceeds as before and this lets us claim that

$$E[F^{\mathcal{A}}] \leq 2(1+\varepsilon^{-1})F^{\texttt{OPT}}$$

6. ACKNOWLEDGEMENTS

We would like to thank Pushkar Tripathi and Ashish Sangwan for many useful discussions.

7. REFERENCES

[1] S. Anand, Naveen Garg, and Amit Kumar. An $O(\log P)$ competitive algorithm for minimizing flow time on related machines. *Unpublished manuscript*, 2009.

[2] Nir Avrahami and Yossi Azar. Minimizing total flow time and total completion time with immediate dispatching. *Algorithmica*, 47(3):253–268, 2007.

[3] Baruch Awerbuch, Yossi Azar, Stefano Leonardi, and Oded Regev. Minimizing the flow time without migration. In *ACM Symposium on Theory of Computing*, pages 198–205, 1999.

[4] Baruch Awerbuch, Yossi Azar, Stefano Leonardi, and Oded Regev. Minimizing the flow time without migration. *SIAM J. Comput.*, 31(5):1370–1382, 2002.

[5] N. Bansal and K. Pruhs. Server scheduling in the L_p norm: A rising tide lifts all boats. In *ACM Symposium on Theory of Computing*, pages 242–250, 2003.

[6] Nikhil Bansal, Ho-Leung Chan, Rohit Khandekar, Kirk Pruhs, Clifford Stein, and Baruch Schieber. Non-preemptive min-sum scheduling with resource augmentation. In *IEEE Symposium on Foundations of Computer Science*, pages 614–624, 2007.

[7] Luca Becchetti and Stefano Leonardi. Nonclairvoyant scheduling to minimize the total flow time on single and parallel machines. *J. ACM*, 51(4):517–539, 2004.

[8] C. Chekuri, S. Khanna, and A. Zhu. Algorithms for weighted flow time. In *ACM Symposium on Theory of Computing*, pages 84–93. ACM, 2001.

[9] Chandra Chekuri, Ashish Goel, Sanjeev Khanna, and Amit Kumar. Multi-processor scheduling to minimize flow time with epsilon resource augmentation. In *ACM Symposium on Theory of Computing*, pages 363–372, 2004.

[10] Naveen Garg and Amit Kumar. Better algorithms for minimizing average flow-time on related machines. In *ICALP (1)*, pages 181–190, 2006.

[11] Naveen Garg and Amit Kumar. Minimizing average flow time on related machines. In *ACM Symposium on Theory of Computing*, pages 730–738, 2006.

[12] Naveen Garg and Amit Kumar. Minimizing average flow-time : Upper and lower bounds. In *IEEE Symposium on Foundations of Computer Science*, pages 603–613, 2007.

[13] Naveen Garg, Amit Kumar, and Muralidhara V N. Minimizing total flow-time: The unrelated case. In *ISAAC*, pages 424–435, 2008.

[14] Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. In *IEEE Symposium on Foundations of Computer Science*, pages 214–221, 1995.

[15] Hans Kellerer, Thomas Tautenhahn, and Gerhard J. Woeginger. Approximability and nonapproximability results for minimizing total flow time on a single machine. In *ACM Symposium on Theory of Computing*, pages 418–426, 1996.

[16] Stefano Leonardi and Danny Raz. Approximating total flow time on parallel machines. In *ACM Symposium on Theory of Computing*, pages 110–119, 1997.

[17] R. Motwani, S. Phillips, and E. Torng. Non-clairvoyant scheduling. *Theoretical Computer Science*, 130(1):17–47, 1994.