2016 IEEE International Parallel and Distributed Processing Symposium

Partitioned Feasibility Tests for Sporadic Tasks on Heterogeneous Machines

Shaurya Ahuja
Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO 63130
shaurya.ahuja@wustl.edu

Kefu Lu
Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO 63130
kefulu@wustl.edu

Benjamin Moseley
Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO 63130
bmoseley@wustl.edu

***Abstract*—In this paper we consider feasibility tests for partitioned scheduling sporadic tasks on a set of heterogeneous machines with different speeds. Previously a 3-approximate feasibility test was known. The feasibility test is a natural, fast and efficient algorithm which greedily assigns the tasks to machines and uses Earliest-Deadline-First (EDF) on each machine. The algorithm is 3-approximate even when compared against any schedule which need not be partitioned and therefore additionally bounds the loss due to using a partitioned scheduler. In our work, we consider the case where the adversary is required to be partitioned. We show that this natural algorithm has an improved approximate feasibility factor of 2. Building on our techniques, we further improve on the best known approximate algorithm when the adversary is partition and show the algorithm achieves a ratio better than 3. In particular, we show it is 2.98 approximate.**

We then consider the case where the partitioned scheduler is required to use Rate-Monotonic-Scheduling (RMS) on each of the machines. Previously, it was known that this algorithm is 3.41-approximate when compared to a non-partitioned adversary. We improve this to 2.41 when comparing to a partitioned adversary. Further, we build on these ideas to improve the best known result when compared against a partitioned adversary and show the algorithm is a 3.34 approximation.

I. Introduction

Feasibility tests for scheduling sporadic real time tasks is a fundamental challenge in real time systems and, due to this, there is a vast literature on the topic. See [8] for a survey. In a typical scenario, there is a task set τ consisting of n tasks $\tau_1, \tau_2, \ldots \tau_n$. Each task generates an infinite set of jobs that are to be scheduled and each individual job has a deadline. A feasibility test for a task system has two components. The first is a scheduler which determines how the jobs will be scheduled. The second is another algorithm that checks whether the scheduling algorithm will be able to complete all jobs by their deadline or not.

In this work, we consider feasibility tests for implicit-deadline sporadic task systems. In a *sporadic task system*, each task τ_i generates a jobs. Each job has a relative hard deadline p_i time steps after its arrival. It is assumed that the task τ_i cannot generate a second job until at least p_i units after the time it generated the last job. Due to this, sometimes p_i is referred to as the *period* of i. We assume that all jobs can start being generated at time 0. Each job generated by task τ_i has computation (processing) time c_i. There has been a considerable amount of work on sporadic task systems.

In the most basic setting, there is a single machine to scheduled the jobs on. In this setting, it is well known that the scheduling algorithm Earliest-Deadline-First (EDF) can feasibly schedule all of the jobs by their deadline, if any algorithm can. Further, it is known that there is a feasibility test for EDF. The feasibility test states that if the total utilization of all of the tasks is at most 1 then EDF can schedule all of the jobs [15]. The *utilization* of a task is $\frac{c_i}{p_i}$ and the total utilization of the task set is $\sum_{i=1}^{n} \frac{c_i}{p_i}$. Another popular algorithm considered is Rate-Monotonic-Scheduling (RMS). The RMS algorithm gives tasks priorities where the priority of a task is the inverse of its period. The algorithm always schedules the job for the tasks which is assigned the highest priority. This algorithm is highly desirable because it assigns static priorities which means that jobs for one task always have the same relative priority when compared to jobs of other tasks. For a task set of size n, it has been shown that if the total utilization of the tasks is at most $n(2^{1/n} - 1) \geq \ln 2$ then RMS will feasibly schedule all of the jobs by their deadlines [15].

A more challenging scheduling environment is when the tasks are required to be scheduled on a set of machines. When scheduling tasks on multiple machines, the most basic environment is the *identical* machine setting where the tasks can be scheduled on m identical machines. Much previous work in the multiple machine setting has focused on the case where the tasks have to be partitioned across the machine. That is, a *partitioned* scheduler always schedules all of the jobs of any fixed task on exactly one machine. Partitioned scheduling in the identical machine setting is challenging and, unlike the single machine setting, it is known that it is strongly NP-Hard to determine if a set of task set can be feasibly scheduled. If EDF is used to schedule jobs on each individual machine, the problem essentially becomes bin-packing.

Due to the inability to efficiently compute an exact feasibility test in the multiple machine setting, previous work has focused on approximate feasibility tests. A feasibility test is said to be an α-approximation if the test returns 'feasible' when the task set can be scheduled on a set of processors of speed an α factor faster. The test returns "infeasible" if no scheduler can schedule the task set on a set of processors of their original speed. There have been several works that have considered approximate partitioned scheduling on identical machines [4], [5], [7].

1530-2075/16 $31.00 © 2016 IEEE
DOI 10.1109/IPDPS.2016.47

Today, most commercial chip architectures have a collection of identical processors on them. However, it is widely believed that future architectures will consist of heterogeneous processors [17], [16], [14], [13]. Heterogeneous architectures are viewed to be advantageous because they allow processors of variable power to be used to processes specialized tasks. These chips are thought to consist of a large number of low power and lower performance processors for processing less demanding jobs. Then there are also a smaller set of high power and high efficiency processors to process more demanding jobs. Such a system of heterogeneous processors delivers significantly better performance for the same energy cost, an advantage over a system of homogeneous processors. Due to this, there has been an increasing interest in scheduling on machines with heterogeneous processors [12], [10], [9], [1], [6], [2], [3].

One of the most widely studied heterogeneous machine settings is when the processors have different speeds. This is known as the uniform or related machines setting. For this setting it is known that there is a 3-approximate feasibility test for partitioned scheduling of sporadic implicit deadline tasks [2]. This algorithm compares against any schedule which is allowed to migrate the jobs across the machines and, therefore, additionally bounds the loss due to forcing the algorithm to be partitioned as well. It is also known that the problem reduces to bin packing when bins can have different sizes and this implies a $(1+\epsilon)$-approximate feasibility test for any $\epsilon > 0$ [11]. This approximation algorithm compares against an adversary which also must be partitioned. Unfortunately, the algorithm for the $(1+\epsilon)$-approximation is quite complicated and the running time depends exponentially on $\frac{1}{\epsilon}$ making it difficult to use in practice.

The work of [2] gave a simple and elegant feasibility test for heterogenous machines, which is quite practical. The algorithm first sorts the processors in increasing order of their speed and then sorts the tasks in decreasing order of their utilization. Then, in this order, the algorithm performs first-fit packing. Each machine runs EDF for the tasks assigned to it. If ever the algorithm is unable to assign a task to a machine (the utilization of all machines exceeds 3 when attempting to assign the task) then it is guaranteed that the task set is infeasible on processors with the original speeds. The same partitioning strategy has been used when each machine runs RMS and a 3.41-approximate feasibility test was shown [3].

For this line of work, several questions loom. The previous work has compared against an adversary which is not required to be partitioned. This, perhaps, is an unfair advantage given to the adversary. In particular, it is more revealing on the true loss to compare to an adversary which is also forced to be partitioned. The main question is, what can be shown when the adversary is required to be partitioned? Can 3 be beaten for EDF or 3.41 for RMS? Further, what is the best possible approximation factor that can be shown for a natural algorithm when the adversary is possibly non-partitioned?

Results: In this work, we present feasibility tests for partitioned scheduling of sporadic tasks with implicit deadlines on heterogenous processors. The focus of our algorithms is to keep the algorithms fast, efficient and easily implementable. Our work begins by improving on the results in [2] and [3]. In particular, the main question we desire to answer, is what can be shown for an efficient practical algorithm when compared against an adversary which is required to be partitioned.

We begin by considering the same algorithm of [2] where each machine runs EDF. We show the following improve result.

Theorem I.1. *There exists a 2-approximate partitioning feasibility test that uses EDF on each machine and runs in $O(nm)$ time. The approximation ratio is relative to a partitioned optimal solution.*

Building on these result, we give a feasibility test when Rate-Monotonic-Scheduling is used on each machine. Again we consider a non-partitioned adversary.

Theorem I.2. *There exists a 2.41-approximate partitioning feasibility test that uses RMS on each machine and runs in $O(nm)$ time. The approximation ratio is relative to a possibly non-partitioned optimal solution.*

Then we try to answer whether or not the results of [2] and [3] are the best possible when comparing against an optimal solution which need not be partitioned. We are able to improve on both of their results by doing a careful analysis, while keeping the algorithm the same. First we consider when EDF is used on each machine.

Theorem I.3. *There exists a 2.98-approximate partitioning feasibility test that uses EDF on each machine and runs in $O(nm)$ time. The approximation ratio is relative to a possibly non-partitioned optimal solution.*

Finally we consider the case where each machine is required to use RMS when compared to a non-partitioned adversary.

Theorem I.4. *There exists a 3.34-approximate partitioning feasibility test that uses RMS on each machine and runs in $O(nm)$ time. The approximation ratio is relative to a possibly non-partitioned optimal solution.*

Our results follow by a careful analysis of the standard linear program (LP) for these scheduling problems. We note that while our analysis relies heavily on the linear program, one need not solve the LP for the feasibility test.

II. Preliminaries

In the problem considered we have a task set τ with n tasks $\tau_1, \tau_2, \ldots \tau_n$. Each task τ_i is sporadic with a relative deadline (or period) of p_i. The jobs generated by any task may start at time 0. Each job for task τ_i has a hard deadline p_i time steps after its arrival. We let $w_i = \frac{c_i}{p_i}$ be the *utilization* of job τ_i if it were scheduled on a unit speed machine. We will refer to this as the utilization of a task. The jobs are to be scheduled on a set M of m machines $m_1, m_2, \ldots, m_m$ of speeds $s_1, s_2, \ldots s_m$. We assume the machines are sorted such that $s_i \leq s_{i+1}$ for all i. The tasks are to be assigned to each of the machines and when a task is assigned to a machine then that machine processes all of the jobs generated by that task. Our algorithm will be given some *speed augmentation* $\alpha \geq 1$. In this case, machine m_i has speed αs_i in the algorithm's schedule.

As observed in previous work [2], a given task system can only be feasibly scheduled on a set of machines if there is a feasible solution to the following natural linear program for the problem. There is a variable $u_{i,j}$ for each job which is intuitively the amount of task i's utilization assigned to machine j. The value of $\frac{u_{i,j}}{w_i}$ denotes the fraction of task i's jobs that are assigned to machine j.

$$\forall i \in \{1,2,\ldots,n\} : \sum_{j=1}^{m} u_{i,j} = \frac{c_i}{p_i} \quad (1)$$

$$\forall i \in \{1,2,\ldots,n\} : \sum_{j=1}^{m} \frac{u_{i,j}}{s_j} \leq 1 \quad (2)$$

$$\forall j \in \{1,2,\ldots,m\} : \sum_{i=1}^{n} \frac{u_{i,j}}{s_j} \leq 1 \quad (3)$$

$$\forall i \in \{1,2,\ldots,n\}, \forall j \in \{1,2,\ldots,m\} : u_{i,j} \geq 0 \quad (4)$$

The first constraint ensures that all tasks are scheduled. The second constraint ensures that no job for the same task is scheduled on more than one machine simultaneously. Finally the third constraint ensures that each machine can process all of the jobs assigned to it. For our analysis, we will use compare to this linear program to determine if a feasible schedule is possible or not. Note that this LP may schedule tasks on multiple machines and is only feasible if some scheduler can feasibly assign the task set allowing migration of jobs between machines. We note that we do not need to explicitly solve the LP, but rather use it only for the analysis.

We now state a useful lemma which was shown in [2], [3]. This lemma helps to bound the total amount tasks must be processed on some of the 'faster' machines in any feasibly LP formation. The full proof can be found in [2] and it follows immediately by the LP formulation.

Lemma II.1 ([2]). *Fix any $\alpha \geq 1$ and any feasible solution $\boldsymbol{u}$ to the LP . Consider any task τ_i and let machine k be any machine where $w_i \leq \alpha s_{k+1}$. Then it is the case that $w_i \leq \frac{\alpha}{\alpha-1} \sum_{j=k+1}^{m} u_{i,j}$.*

To analyze our algorithm the following well known theorems about EDF and RMS will be useful.

Theorem II.2 ([15]). *A set of tasks S can be feasibly scheduled on a machine of speed s by EDF if $\sum_{\tau_i \in S} w_i \leq s$.*

Theorem II.3 ([15]). *A set of tasks S can be feasibly scheduled on a machine of speed s by RMS if $\sum_{\tau_i \in S} w_i \leq |S|(2^{\frac{1}{|S|}} - 1)s$. In particular, S can be feasibly scheduled if $\sum_{\tau_i \in S} w_i \leq (\ln 2)s$.*

III. Algorithms

We now describe our algorithm and say it has some speed augmentation $\alpha \geq 1$. The algorithm uses any algorithm A to schedule tasks that are assigned to a machine. This algorithm will be set to EDF or RMS in the later sections. The algorithm sorts the tasks $\tau_1, \tau_2, \ldots \tau_n$ such that $w_i \geq w_{i+1}$. The algorithm then sorts the machines $m_1, m_2, \ldots, m_m$ in increasing order of their speeds. Then the algorithm runs first fit. A task τ_j can be feasibly scheduled on a machine of speed s that has been assigned a set S of tasks when A is EDF if $\sum_{\tau_i \in S \cup \{\tau_j\}} w_i \leq \alpha s$. Similarly, a task τ_j can be feasibly scheduled on a machine of speed s that has been assigned a set S of tasks when A is RMS if $\sum_{\tau_i \in S \cup \{\tau_j\}} w_i \leq (|S|+1)(2^{\frac{1}{|S|+1}} - 1)\alpha s$. In either case, if there is no such machine the algorithm declares failure. Note that this algorithm runs in $O(n * \log n + nm)$ time, since tasks and machines also need to be sorted.

```
1: Sort the tasks tasks $\tau_1, \tau_2, \ldots \tau_n$ such that $w_i \geq w_{i+1}$
2: Sort the machines $m_1, m_2, \ldots, m_m$ such that $s_i \leq s_{i+1}$
3: for $i = 1$ to $n$ do
4:     Assign $\tau_i$ to the machine $m_j$ for the smallest $j$ such that the jobs assigned to $m_j$ pass a single machine feasibility test.
5: end for
```

In the following section, we analyze this algorithm when A is EDF and then the following section when A is RMS.

IV. EDF on Related Machines

In this section we analyze the algorithm given in the previous section when A is set to be EDF. In this proof we will first define four constants c_s, c_f, d_f, f_f whose values will be given at a later point. We will assume that our algorithm is given speed augmentation $\alpha \geq 1$. The value of α depends on whether we are comparing against the LP, a non-partitioned adversary, or a partitioned adversary. When the adversary is the LP, we set $\alpha = 2.98$ and, if it is partitioned, we set $\alpha = 2$. In the algorithm's schedule each machine m_i has speed αs_i. The machine has speed s_i in the the optimal schedule we compare against.

Notice that if our algorithm does not declare failure, then all the tasks can be feasible scheduled due to Theorem II.2. Thus, we only need to show that if the algorithm declares failure then the tasks cannot be feasibly scheduled by the adversary without speed augmentation. As mentioned, we will consider both an adversary scheduler which is either an arbitrary schedule or partitioned. For the the arbitrary schedule we compare against the LP given in the previous section and show it must be infeasible. For the partitioned scheduler, we only need to argue that any partitioned scheduler will have some machine of speed s whose utilization is greater than s implying the schedule is infeasible.

Consider any set of tasks and assume the algorithm declares failure. Let the task τ_n be the task for which the algorithm fails. Notice that we can assume that the only tasks are those already scheduled by the algorithm and the task τ_n. Let $\mathcal{S}_i$ be the set of tasks assigned to machine m_i by the algorithm. Notice that since that algorithm declared failure, for any machine m_i it is the case that $\sum_{\tau_i \in \mathcal{S}_i \cup \{\tau_n\}} w_i > \alpha s$ by definition of the algorithm. That is, we cannot feasible assign τ_n to any machine.

For analysis we order the machines $m_1 \ldots m_m$ such that the speed of the machines obey $s_1 \leq s_2 \leq \ldots \leq s_m$. We define the *fast* machines to be the set M_f where any machine has speed at least s_f where we define $\alpha s_f = w_n c_s$ using the first constant. Similarly, we define the *slow* machines to be the set M_s where any machine has speed less than s_s where $\alpha s_s = w_n$. The machines with speed between s_s and s_f we group as the *medium* machines M_m. Clearly $M = M_f \cup M_m \cup M_s$ where M is the set of all machines. Note that because of the ordering of the machines by speed, these sets are contiguous. Therefore $\{m_1, \ldots, m_k\}$, $\{m_{k+1}, \ldots, m_{j-1}\}$ and $\{m_j, \ldots, m_m\}$ are the members of the slow, medium, and fast machines respectively.

Now we are ready to divide our proof into two cases depending on whether the aggregate speed of the fast machines is large or small compared to the overall total speed of all machines. The bulk of the analysis is for the case where the adversary is non-partitioned, but along the way we will be able to bound the performance against a partitioned scheduler.

A. Powerful Fast Machines

In this section, we will show that the task set cannot be feasible for any algorithm if $\sum_{m_\ell \in M_f} \alpha s_\ell > \frac{1}{c_f} \sum_{m_\ell \in (M_m \cup M_f)} \alpha s_\ell$. This is effectively the case where the aggregate speed of the fast machines is large compared to the total speed of all machines. For this first section the initial two constants are used heavily. The values of the two will be established later to be $c_s = 2.868$ and $c_f = 28.412$. The rest of the section will be devoted to proving the main lemma:

Lemma IV.1. *If $\sum_{m_\ell \in M_f} \alpha s_\ell > \frac{\alpha}{c_f} \sum_{m_\ell \in (M_m \cup M_f)} s_\ell$ and our algorithm declares failure then there is no feasible solution to the LP when $\alpha \geq 2.98$.*

To simplify notation we will define a function $S(M') = \sum_{m_\ell \in M'} s_\ell$ on a set of machines is the sum of the speed of all the machines in M'. Now assume for the sake of contradiction that the lemma is not true, that is, suppose $\alpha S(M_f) > \frac{\alpha}{c_f} S(M_m \cup M_f)$ and there is a feasible solution to the LP.

In our algorithm, τ_n cannot be assigned to any of the machines in M_s even if they have no tasks because those machines are too slow (recall the tasks are partitioned in our algorithm). We now consider the other two sets.

Medium Machines: Knowing that the algorithm reported failure, there is at least one task on each machine in M_m, otherwise the algorithm could assign τ_n to an empty machine $m_\ell \in M_m$ with speed $\alpha s_\ell \geq w_i$. Furthermore, consider a specific machine $m_\ell \in M_m$. We argue that $\sum_{\tau_i \in \mathcal{S}_\ell} w_i \geq \frac{1}{2}\alpha s_\ell$. Suppose this is not true, then $w_n > \frac{1}{2}\alpha s_\ell$ since the algorithm fails to assign τ_n to m_ℓ. However, recall that all the assigned tasks τ_i have $w_i \geq w_n$ and that there is at least one task assigned to m_ℓ. Therefore, m_ℓ has a task assigned to it. This is a contradiction since $w_i \geq w_n \geq \frac{1}{2}\alpha s_\ell$. This is true for every machine in M_m therefore $\sum_{m_\ell \in M_m} \sum_{\tau_i \in \mathcal{S}_\ell} w_i \geq \frac{\alpha}{2} S(M_m)$.

Fast Machines: By definition, these machines have speed at least $\alpha w_n c_s$. Again we consider a machine $m_\ell \in M_f$. We will show that $\sum_{\tau_i \in \mathcal{S}_\ell} w_i \geq (1 - \frac{1}{c_s})\alpha s_\ell$. For the sake of contradiction, if this is not so, then notice that $\sum_{\tau_i \in \mathcal{S}_\ell} w_i \leq (1 - \frac{1}{c_s})\alpha s_\ell$. However, then the algorithm could feasibly assign task τ_n to machine m_ℓ, a contradiction. This is again true for every machine in M_f so therefore $\sum_{m_\ell \in M_f} \sum_{\tau_i \in \mathcal{S}_\ell} w_i \geq (1 - \frac{1}{c_s})\alpha S(M_f)$.

Together, the previous two arguments imply.

Lemma IV.2. $\sum_{\ell=k+1}^{j} \frac{\alpha}{2} s_\ell + \sum_{\ell=j}^{m} (1 - \frac{1}{c_s})\alpha s_\ell < \sum_{i=1}^{n-1} w_i$

Proof: The lemma directly follows from combining the analysis of the two groups. ■

Now we are ready to relate the utilization of the jobs to the aggregate speeds of the machines. Note that the following corollary holds true even if the fast group contains very few or even no machines (i.e $j = m$). The corollary will be useful for the other case where there are not many fast machines.

Corollary IV.3. $\frac{\alpha}{2} \sum_{\ell=k+1}^{m} s_\ell \leq \sum_{i=1}^{n-1} w_i$

Proof: Note that since $c_s > 2$ this corollary is straightforward given the previous lemma. ■

Now before we prove Lemma IV.1, we can show Theorem I.1. The previous corollary shows that the sum of the speeds of the medium and fast machines are less than $\sum_{i=1}^{n-1} w_i$. We know our algorithm cannot schedule any of the tasks on a machine m_ℓ where $\ell \leq k$ because the utilization would be greater than αs_ℓ. In this case, a partitioned adversary cannot as well. Thus, by setting $\alpha = 2$, the previous corollary implies that a partitioned advisory also cannot schedule the tasks and we get Theorem I.1. Note that we will choose a larger α when considering a non-partitioned adversary.

Proof of Lemma IV.1 (main) We will arrive at a contradiction by proving that the tasks which are assigned before τ_n will necessary cause there to be no solution to the LP. We set $\alpha \geq 2.98$ and we use the analysis of the previous two sets of machines to bound the utilization requirements of the tasks. Recall by assumption that the fast machines have more than a fraction $\frac{1}{c_f}$ of the total speed of the medium and the fast sets together. Also note that the speed of the machines in the LP are an $\frac{1}{\alpha}$ factor slower than the algorithm.

$$\begin{aligned}
\sum_{i=1}^{n-1} w_i &> \sum_{\ell=k+1}^{j} \frac{1}{2}\alpha s_\ell + \sum_{\ell=j}^{m}(1 - \frac{1}{c_s})\alpha s_\ell \quad (\textit{Lemma IV.2}) \\
&\geq \sum_{\ell=k+1}^{m} \frac{1}{2}\alpha s_\ell + \sum_{\ell=j}^{m}(\frac{1}{2} - \frac{1}{c_s})\alpha s_\ell \\
&> \frac{1}{2}\sum_{\ell=k+1}^{m} \alpha s_\ell + \frac{1}{c_f}(\frac{1}{2} - \frac{1}{c_s}) \sum_{\ell=k+1}^{m} \alpha s_\ell \\
\sum_{i=1}^{n-1} w_i &> \left(\frac{1}{2} + \frac{1}{2c_f} - \frac{1}{c_s c_f}\right) \sum_{\ell=k+1}^{m} \alpha s_\ell
\end{aligned}$$

We then invoke Lemma II.1 on the left side of the equation for any feasible solution to the LP $\mathbf{u}$.

$$\sum_{i=1}^{n-1} \frac{\alpha}{\alpha-1} \sum_{\ell=k+1}^{m} u_{i,\ell} > \alpha \left(\frac{1}{2} + \frac{1}{2c_f} - \frac{1}{c_s c_f} \right) \sum_{\ell=k+1}^{m} s_\ell$$

$$\frac{1}{\alpha-1} \sum_{i=1}^{n-1} \sum_{\ell=k+1}^{m} u_{i,\ell} > \left(\frac{1}{2} + \frac{1}{2c_f} - \frac{1}{c_s c_f} \right) \sum_{\ell=k+1}^{m} s_\ell$$

$$\sum_{i=1}^{n-1} \sum_{\ell=k+1}^{m} u_{i,\ell} > (\alpha-1) \left(\frac{1}{2} + \frac{1}{2c_f} - \frac{1}{c_s c_f} \right) \sum_{\ell=k+1}^{m} s_\ell$$

From the choice of constants we obtain that $(\alpha - 1)\left(\frac{1}{2} + \frac{1}{2c_f} - \frac{1}{c_s c_f}\right) \approx 1.005 > 1$ which implies:

$$\sum_{i=1}^{n-1} \sum_{\ell=k+1}^{m} u_{i,\ell} > \sum_{\ell=k+1}^{m} s_\ell \tag{5}$$

This requires that for some machine, $\sum_{i=1}^{n-1} u_{i,\ell} > s'_\ell$. Rewriting this contradicts one of the LP conditions: $\sum_{i=1}^{n-1} \frac{u_{i,\ell}}{s'_\ell} > 1$. This directly means that no scheduling algorithm can feasibly schedule the set of tasks. □

B. Powerful Slow Machines

In this section, we will show that the task set cannot be feasible for any algorithm if $\sum_{m_\ell \in M_f} \alpha s_\ell \leq \frac{1}{c_f} \sum_{m_\ell \in (M_m \cup M_f)} \alpha s_\ell$. This is effectively the case where the aggregate speed of the fast machines is small compared to the total speed of all machines. In this section two more constants are used, f_w and f_f. We will later establish that $f_w = 0.811$ and $f_f = 0.125$. Importantly we have $0 \leq f_w \leq 1$ and $0 \leq f_f \leq 1$. Recall we have defined the notion of *fast* machines in our algorithm's schedule. Note that such a fast machine in the LP only has $\frac{1}{\alpha}$ of the speed. The rest of this section will be devoted to proving the following main lemma.

Lemma IV.4. *If $\sum_{m_\ell \in M_f} \alpha s_\ell \leq \frac{1}{c_f} \sum_{m_\ell \in (M_m \cup M_f)} \alpha s_\ell$ and our algorithm fails in scheduling the task set, then the LP is infeasible.*

Let τ be the set of all tasks. Partition τ such that $\tau = \mathcal{S}_f \cup \mathcal{S}_s$ where a task τ_i is in $\mathcal{S}_s$ if for that task less than f_f fraction of the task is processed by the fast machines LP. In other words, $\sum_{m_\ell \in M_f} u_{i,\ell} < f_f w_i$.

Lemma IV.5. $\sum_{\tau_i \in \mathcal{S}_s} w_i > f_w \sum_{\tau_i \in T} w_i$

Proof: To simplify notation define $W(\mathcal{S}) = \sum_{\tau_i \in \mathcal{S}} w_i$ for any set $\mathcal{S}$ of tasks. We can rewrite the lemma as $W(\mathcal{S}_s) > f_w W(T)$. Then, for the sake of contradiction, suppose that $W(\mathcal{S}_s) \leq f_w W(T)$. By definition of $\mathcal{S}_s$ we know that $\mathcal{S}_f = T \setminus \mathcal{S}_s$ and so $W(\mathcal{S}_f) > W(T) - W(\mathcal{S}_s) = (1 - f_w)W(T)$.

Therefore $\sum_{i=1}^{n} \sum_{m_\ell \in M_f} u_{i,\ell}$ is at least:

$$\begin{aligned} f_f W(S_f) &> f_f(1-f_w)W(T) \\ &> \frac{f_f(1-f_w)}{2} \sum_{\ell=k+1}^{m} \alpha s_\ell \quad (Corollary\ IV.3) \\ &\geq \frac{c_f f_f(1-f_w)}{2} \sum_{m_\ell \in M_f} \alpha s_\ell \quad (Lemma\ IV.1) \\ &\geq \frac{\alpha c_f f_f(1-f_w)}{2} \sum_{m_\ell \in M_f} s_\ell \end{aligned}$$

Note that $\frac{\alpha c_f f_f(1-f_w)}{2} > 1$ which violates a constraint in the LP, a contradiction to there being a feasible solution to the LP. ■

Consider any task $\tau_i \in \mathcal{S}_s$. Let $f_{i,m} = \sum_{m_\ell \in M_m} u_{i,\ell}$ be the fraction of the task which is processed on machines in M_m. Recall that the medium machines have speed at most αs_f while the slow machines have speed at most αs_s. Our goal is to bound the amount τ_i is processed on the slow machines in the LP. First we derive an inequality based on the LP.

Lemma IV.6. *For any task τ_i where $i \neq n$ we have $\frac{(1-f_{i,m}-f_f)}{s_s} w_i + \frac{f_{i,m}}{s_f} w_i \leq 1$.*

Proof: From the LP conditions it is true that $\forall i \in \{1, 2, \ldots, n\} : \sum_{\ell=1}^{m} \frac{u_{i,\ell}}{s_\ell} \leq 1$. We rearrange the LP condition to produce:

$$\sum_{\ell=1}^{m} \frac{u_{i,\ell}}{s_\ell} \leq 1$$

$$\sum_{m_\ell \in M_s} \frac{u_{i,\ell}}{s_\ell} + \sum_{m_\ell \in M_m} \frac{u_{i,\ell}}{s_\ell} + \sum_{m_\ell \in M_s} \frac{u_{i,\ell}}{s_\ell} \leq 1$$

$$\frac{1}{s_s} \sum_{m_\ell \in M_s} u_{i,\ell} + \frac{1}{s_f} \sum_{m_\ell \in M_m} u_{i,\ell} \leq 1$$

By definition, $\sum_{m_\ell \in M_s} u_{i,\ell} \geq (1 - f_{i,m} - f_f)w_i$ and $w_i f_{i,m} = \sum_{m_\ell \in M_m} u_{i,\ell}$. Substitution of these terms for the summations we obtain the lemma. ■

Now we can bound $f_{i,m}$.

Lemma IV.7. *For any task τ_i where $i \neq n$ we have $f_{i,m} \geq \frac{1+\alpha f_f - \alpha}{\alpha(\frac{1}{c_s} - 1)}$.*

Proof: Recall that $\alpha s_s = w_i$ and $\alpha s_f = w_i c_s$. From substitution into Lemma IV.6 we obtain:

$$\alpha(1 - f_{i,m} - f_f) + \frac{\alpha f_{i,m}}{c_s} \leq 1$$

$$\alpha - \alpha f_{i,m} - \alpha f_f + \alpha \frac{f_{i,m}}{c_s} \leq 1$$

$$\alpha f_{i,m}\left(\frac{1}{c_s} - 1\right) \leq 1 + \alpha f_f - \alpha$$

$$f_{i,m} \geq \frac{1 + \alpha f_f - \alpha}{\alpha(\frac{1}{c_s} - 1)} \quad [\text{Since } \; c_s > 1]$$

■

Proof of Lemma IV.4 (main) We have that $\sum_{\tau_i \in T} w_i > \frac{1}{2} \sum_{\ell=k+1}^{m} s_\ell \geq \frac{1}{2} \sum_{m_\ell \in M_m} s_\ell$ (Corollary IV.3). Now consider all the tasks in $\mathcal{S}_s$. By Lemma IV.5 their total utilization is at least $f_w \sum_{i \in T} w_i$, hence $\sum_{\tau_i \in \mathcal{S}_s} w_i > \frac{f_w}{2} \sum_{m_\ell \in M_m} s_\ell$. We have defined that the LP does at least a $f_{i,m}$ fraction of each of these tasks on the medium machines. Therefore, using Lemma IV.7 we bound the amount of work the LP must do on the medium machines by $f_{i,m}(\sum_{\tau_i \in \mathcal{S}_s} w_i) > \frac{f_{i,m} f_w}{2} \sum_{m_\ell \in M_m} \alpha s_\ell > \sum_{m_\ell \in M_m} s_\ell$ which is a contradiction. □

Thus, Lemmas IV.2 and IV.5 prove Theorem I.3.

V. Rate-Monotonic on Related Machines

In this section we analyze the algorithm when A is set to be RMS. This means our algorithm uses RMS on each machine to decide if a task set is schedulable or not. According to the algorithm, the tasks are sorted in order of non-increasing utilization, the machines are sorted in order of increasing speed and tasks are assigned to the first machine which passes the feasibility test. The performance of our algorithm is given by Theorem I.4, which is the main theorem for RMS. This section is devoted to proving the main theorem, along the way we will also prove Theorem I.2.

For the sake of contradiction we assume the main theorem is false. This means that for some task set τ, our algorithm declares failure on the heterogeneous platform, despite there existing a solution for either partitioned scheduler or the LP.

When our algorithm declares failure for a given task set τ, then there must have been a particular task τ_n caused the failure. All tasks with index greater than τ_n do not affect the schedulability of τ_n and therefore we remove them from consideration.

Similar to the proof on EDF, we introduce four constants, c_s, c_f, d_f, f_f whose values will be given at a later point. We group the machines into the slow, medium and fast groups. Let machines $\{m_1, \ldots, m_k\}$ be the slow machines with speed less than s_s where $\alpha s_s = w_n$. Further, the fast machines $\{m_j, \ldots, m_m\}$ are the one with speed at least s_f, with $\alpha s_f = w_n c_s$. The medium machines are the machines $\{m_{k+1}, \ldots, m_{j-1}\}$ which have speeds between that of the slow and fast groups.

We again divide our analysis into two cases depending on the aggregate speed of the fast machines.

A. Fast Machines

In this section, we will show that the task set cannot be feasible for any algorithm if $\sum_{m_\ell \in M_f} \alpha s_\ell > \frac{1}{c_f} \sum_{m_\ell \in (M_m \cup M_f)} \alpha s_\ell$. This is when the aggregate speed of the fast machines is large. The value of the constants used in this section will be established as $c_s = 2.00$ and $c_f = 13.25$. This will be formally stated in the following lemma.

Lemma V.1. *If $\sum_{m_\ell \in M_f} \alpha s_\ell > \frac{\alpha}{c_f} \sum_{m_\ell \in (M_m \cup M_f)} s_\ell$ and our algorithm declares failure then there is no feasible solution to the LP when $\alpha \geq 3.34$.*

We will prove this lemma by contradiction, that is, suppose there still is a feasible solution to the LP given the condition in the lemma. For contradiction our algorithm still failed. We will prove some facts about the load on the fast and medium machines.

Lemma V.2. *If our algorithm declares failure, then for any fast machine m_l with assigned task set $\mathcal{S}_l$, the total utilization of the tasks is at least $\sum_{\tau_i \in \mathcal{S}_l} w_i > (\ln(2) - \frac{1}{c_s})\alpha s_f$.*

Proof: Consider a machine, m_l which is fast. Because the algorithm failed, we know that task τ_n cannot be assigned to a machine m_l. Let $\mathcal{S}_l$ be the set of tasks already assigned on m_l. Now, using the rate-monotonic bound we have

$$w_n + \sum_{\tau_i \in \mathcal{S}_l} w_i > (n_j + 1)(2^{\frac{1}{n_j+1}} - 1)\alpha s_f \geq \ln(2)\alpha s_f$$

For ease of notation consider $k = \ln(2) - \frac{1}{c_s}$. For contradiction, let us assume that the fast machines are utilized less than $k\alpha s_f$. Consider the total utilization of tasks on machine m_l if τ_n were to be added.

$$\begin{aligned} w_n + \sum_{\tau_i in \mathcal{S}_l} w_i < w_n + k\alpha s_f &= \frac{1}{c_s} s_f + (\ln(2) - \frac{1}{c_s})\alpha s_f \\ &= \ln(2)\alpha s_f \end{aligned}$$

This is a contradiction with the result given by the rate-monotonic bound. ■

Lemma V.3. *If our algorithm declares failure, then all machines with at least αs_s-speed are loaded to at least $(\sqrt{2} - 1)\alpha s_s$*

Proof: Since our algorithm declared failure, therefore it could not fit task τ_n on any of machines with speed at least αs_s. Consider any machine m with speed s for which the algorithm could not assign to and assume for contradiction that the total utilization on the machine is less than $(\sqrt{2}-1)s$. Recall that we know $s \geq \alpha s_s$, therefore, $w_n < s$. Hence, for our algorithm to have declared failure, there must have been at least one task already assigned to m as otherwise τ_n can be assigned to m. Suppose that there are $k \geq 1$ tasks already assigned to m, now we consider assigning task τ_n to m. Note that the total load on the machine is currently less than $(\sqrt{2} - 1)s$.

Since there are k tasks already on the machine with a total utilization of less than $(\sqrt{2}-1)s$, there is at least one task has utility of no more than $\frac{(\sqrt{2}-1)s}{k}$. We know that τ_n's utilization is bounded by this task due to the algorithm's ordering. Now, according to II.3, RMS on a single machine should not fail unless the total utilization is at least $(k+1)(2^{\frac{1}{k+1}} - 1)$, recall that there are $k+1$ tasks on the machine if τ_n were to be assigned to it. Therefore, we compare the total utilization on m after assigning task τ_n to m to this quantity.

$$\begin{aligned} (\sqrt{2} - 1)s + \frac{(\sqrt{2} - 1)s}{k} &= \frac{k+1}{k}(\sqrt{2} - 1) \\ &\leq (k+1)(2^{\frac{1}{k+1}} - 1) \end{aligned}$$

Which is true for all $k \geq 1$. Therefore, it is possible for the algorithm to assign the task to machine m, and this is a contradiction to our algorithm declaring failure. ■

Before moving on to prove the main lemma of this section, we first show Theorem I.2. We know our algorithm cannot schedule any of the tasks on a machine m_ℓ where $\ell \leq k$ because the utilization would be greater than αs_ℓ. In this case, a partitioned adversary cannot as well. Thus, by setting $\alpha = \frac{1}{\sqrt{2}-1} \approx 2.41$, the previous corollary implies that a partitioned advisory also cannot schedule the tasks and we get Theorem I.2. Note that we will choose a larger α when considering a non-partitioned adversary.

Now we combine the bound on the two groups in order to prove the main lemma of this subsection.

Proof of Lemma V.1 We will arrive at a contradiction by proving that the tasks which are assigned before task τ_n will necessary cause there to be no solution to the LP. We set $\alpha \geq 3.33$ and we use the analysis of the previous two sets of machines to bound the utilization requirements of the tasks. Recall by assumption that the fast machines have more than a fraction $\frac{1}{c_f}$ of the total speed of the medium and the fast sets together. Also note that the speed of the machines in the LP are an $\frac{1}{\alpha}$ factor slower than the algorithm.

$$\sum_{i=1}^{n-1} w_i > \sum_{\ell=k+1}^{j} (\sqrt{2}-1)\alpha s_\ell + \sum_{\ell=j}^{m} (\ln(2) - \frac{1}{c_s})\alpha s_\ell$$
$$> (\sqrt{2}-1) \sum_{\ell=k+1}^{m} \alpha s_\ell + \frac{1}{c_f}(\ln(2) - \frac{1}{c_s}) \sum_{\ell=k+1}^{m} \alpha s_\ell$$
$$\sum_{i=1}^{n-1} w_i > \left(\sqrt{2}-1 + \frac{1}{c_f}(\ln(2) - \frac{1}{c_s})\right) \sum_{\ell=k+1}^{m} \alpha s_\ell$$

We then invoke Lemma II.1 on the left side of the equation for any feasible solution to the LP **u**.

$$\sum_{i=1}^{n-1} \frac{\alpha}{\alpha-1} \sum_{\ell=k+1}^{m} u_{i,\ell} > \alpha \left(\sqrt{2}-1 + \frac{1}{c_f}(\ln(2) - \frac{1}{c_s})\right) \sum_{\ell=k+1}^{m} s_\ell$$
$$\frac{1}{\alpha-1} \sum_{i=1}^{n-1} \sum_{\ell=k+1}^{m} u_{i,\ell} > \left(\sqrt{2}-1 + \frac{1}{c_f}(\ln(2) - \frac{1}{c_s})\right) \sum_{\ell=k+1}^{m} s_\ell$$
$$\sum_{i=1}^{n-1} \sum_{\ell=k+1}^{m} u_{i,\ell} >$$
$$(\alpha-1) \left(\sqrt{2}-1 + \frac{1}{c_f}(\ln(2) - \frac{1}{c_s})\right) \sum_{\ell=k+1}^{m} s_\ell$$

From the choice of constants we obtain that $(\alpha - 1)\left(\sqrt{2}-1 + \frac{1}{c_f}(\ln(2) - \frac{1}{c_s})\right) \approx 1.004 > 1$ which implies:

$$\sum_{i=1}^{n-1} \sum_{\ell=k+1}^{m} u_{i,\ell} > \sum_{\ell=k+1}^{m} s_\ell$$

This requires that for some machine, $\sum_{i=1}^{n-1} u_{i,\ell} > s'_\ell$. Rewriting this contradicts one of the LP conditions: $\sum_{i=1}^{n-1} \frac{u_{i,\ell}}{s'_\ell} > 1$. This directly means that no scheduling algorithm can feasibly schedule the set of tasks. □

B. Slow Machines

In this section, we will show that the task set cannot be feasible for any algorithm if $\sum_{m_\ell \in M_f} \alpha s_\ell \leq \frac{1}{c_f} \sum_{m_\ell \in (M_m \cup M_f)} \alpha s_\ell$. This is effectively the case where the aggregate speed of the fast machines is small compared to the total speed of all machines. In this section two more constants are used, f_w and f_f. We will later establish that $f_w = 0.72$ and $f_f = 0.1956$. Importantly we have $0 \leq f_w \leq 1$ and $0 \leq f_f \leq 1$. Recall we have defined the notion of *fast* machines in our algorithm's schedule. Note that such a fast machine in the LP only has $\frac{1}{\alpha}$ of the speed. The rest of this section will be devoted to proving the following main lemma.

Lemma V.4. *If $\sum_{m_\ell \in M_f} \alpha s_\ell \leq \frac{1}{c_f} \sum_{m_\ell \in (M_m \cup M_f)} \alpha s_\ell$ and our algorithm fails in scheduling the task set, then the LP is infeasible.*

We will now argue about the solution of the LP. Let τ be the set of all tasks. Partition τ such that $\tau = \mathcal{S}_f \cup \mathcal{S}_s$ where a task τ_i is in $\mathcal{S}_s$ if for that task less than f_f fraction of the task is processed by the fast machines LP. In other words, $\sum_{m_\ell \in M_f} u_{i,\ell} < f_f w_i$.

Lemma V.5. $\sum_{\tau_i \in \mathcal{S}_s} w_i > f_w \sum_{\tau_i \in T} w_i$

Proof: To simplify notation define $W(\mathcal{S}) = \sum_{\tau_i \in \mathcal{S}} w_i$ for any set $\mathcal{S}$ of tasks. We can rewrite the lemma as $W(\mathcal{S}_s) > f_w W(T)$. Then, for the sake of contradiction, suppose that $W(\mathcal{S}_s) \leq f_w W(T)$. By definition of $\mathcal{S}_s$ we know that $\mathcal{S}_f = T \setminus \mathcal{S}_s$ and so $W(\mathcal{S}_f) > W(T) - W(\mathcal{S}_s) = (1 - f_w)W(T)$. Therefore $\sum_{i=1}^{n} \sum_{m_\ell \in M_f} u_{i,\ell}$ is at least:

$$f_f W(S_f) > f_f(1-f_w)W(T)$$
$$> (\sqrt{2}-1) f_f(1-f_w) \sum_{\ell=k+1}^{m} \alpha s_\ell \quad (Lemma\ V.3)$$
$$\geq (\sqrt{2}-1) c_f f_f(1-f_w) \sum_{m_\ell \in M_f} \alpha s_\ell$$
$$\geq (\sqrt{2}-1) \alpha c_f f_f(1-f_w) \sum_{m_\ell \in M_f} s_\ell$$

Note that $(\sqrt{2}-1)\alpha c_f f_f(1-f_w) \approxeq 1.003 > 1$ which violates a constraint in the LP, a contradiction to there being a feasible solution to the LP. ■

NOTE: Below is an argument only on the LP, so it is the same in both RMS and EDF.

Now we argue about the maximum amount of processing a slow machine can do in the LP. Consider any task $\tau_i \in \mathcal{S}_s$. Let $f_{i,m} = \sum_{m_\ell \in M_m} u_{i,\ell}$ be the fraction of the task which is processed on machines in M_m. Recall that the medium machines have speed at most αs_f while the slow machines have speed at most αs_s. Our goal is to bound the amount task τ_i is processed on the slow machines in the LP. First we derive an inequality based on the LP.

Lemma V.6. *For any task τ_i where $i \neq n$ we have $\frac{(1-f_{i,m}-f_f)}{s_s} w_i + \frac{f_{i,m}}{s_f} w_i \leq 1$.*

Proof: From the LP conditions it is true that $\forall i \in \{1, 2, \ldots, n\} : \sum_{\ell=1}^{m} \frac{u_{i,\ell}}{s_\ell} \leq 1$. We rearrange the LP condition to produce:

$$\sum_{\ell=1}^{m} \frac{u_{i,\ell}}{s_\ell} \leq 1$$

$$\sum_{m_\ell \in M_s} \frac{u_{i,\ell}}{s_\ell} + \sum_{m_\ell \in M_m} \frac{u_{i,\ell}}{s_\ell} + \sum_{m_\ell \in M_s} \frac{u_{i,\ell}}{s_\ell} \leq 1$$

$$\frac{1}{s_s} \sum_{m_\ell \in M_s} u_{i,\ell} + \frac{1}{s_f} \sum_{m_\ell \in M_m} u_{i,\ell} \leq 1$$

By definition, $\sum_{m_\ell \in M_s} u_{i,\ell} \geq (1 - f_{i,m} - f_f) w_i$ and $w_i f_{i,m} = \sum_{m_\ell \in M_m} u_{i,\ell}$. Substitution of these terms for the summations we obtain the lemma. ■

Now we can bound $f_{i,m}$.

Lemma V.7. *For any task τ_i where $i \neq n$ we have $f_{i,m} \geq \frac{1+\alpha f_f - \alpha}{\alpha(\frac{1}{c_s}-1)}$.*

Proof: Recall that $\alpha s_s = w_i$ and $\alpha s_f = w_i c_s$. From substitution into Lemma V.6 we obtain:

$$\alpha(1 - f_{i,m} - f_f) + \frac{\alpha f_{i,m}}{c_s} \leq 1$$

$$\alpha - \alpha f_{i,m} - \alpha f_f + \alpha \frac{f_{i,m}}{c_s} \leq 1$$

$$\alpha f_{i,m}(\frac{1}{c_s} - 1) \leq 1 + \alpha f_f - \alpha$$

$$f_{i,m} \geq \frac{1 + \alpha f_f - \alpha}{\alpha(\frac{1}{c_s} - 1)} \quad [\text{Since } \; c_s > 1]$$

■

Proof of Lemma V.4 (main) We have that $\sum_{\tau_i \in T} w_i > (\sqrt{2}-1) \sum_{\ell=k+1}^{m} s_\ell \geq (\sqrt{2}-1) \sum_{m_\ell \in M_m} s_\ell$. Now consider all the tasks in $\mathcal{S}_s$. By Lemma V.5 their total utilization is at least $f_w \sum_{i \in T} w_i$, hence $\sum_{\tau_i \in \mathcal{S}_s} w_i > (\sqrt{2}-1) f_w \sum_{m_\ell \in M_m} s_\ell$. We have defined that the LP does at least a $f_{i,m}$ fraction of each of these tasks on the medium machines. Therefore, using Lemma V.7 we bound the amount of work the LP must do on the medium machines by $f_{i,m}(\sum_{\tau_i \in \mathcal{S}_s} w_i) > (\sqrt{2}-1) f_{i,m} f_w \sum_{m_\ell \in M_m} \alpha s_\ell > \sum_{m_\ell \in M_m} s_\ell$ which is a contradiction. □

Thus, Lemmas V.1 and V.4 prove Theorem I.4.

References

[1] S. Anand, Naveen Garg, and Amit Kumar. Resource augmentation for weighted flow-time explained by dual fitting. In *SODA*, pages 1228–1241, 2012.

[2] Björn Andersson and Eduardo Tovar. Competitive analysis of partitioned scheduling on uniform multiprocessors. In *21th International Parallel and Distributed Processing Symposium (IPDPS 2007), Proceedings, 26-30 March 2007, Long Beach, California, USA*, pages 1–8, 2007.

[3] Björn Andersson and Eduardo Tovar. Competitive analysis of static-priority partitioned scheduling on uniform multiprocessors. In *13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007), 21-24 August 2007, Daegu, Korea*, pages 111–119, 2007.

[4] Sanjoy K. Baruah and Nathan Fisher. The partitioned multiprocessor scheduling of deadline-constrained sporadic task systems. *IEEE Trans. Computers*, 55(7):918–923, 2006.

[5] Sanjoy K. Baruah and Nathan Fisher. The partitioned dynamic-priority scheduling of sporadic task systems. *Real-Time Systems*, 36(3):199–226, 2007.

[6] Jivitej S. Chadha, Naveen Garg, Amit Kumar, and V. N. Muralidhara. A competitive algorithm for minimizing weighted flow time on unrelated machines with speed augmentation. In *Symposium on Theory of Computing*, pages 679–684, 2009.

[7] Jian-Jia Chen and Samarjit Chakraborty. Resource augmentation bounds for approximate demand bound functions. In *Proceedings of the 32nd IEEE Real-Time Systems Symposium, RTSS 2011, Vienna, Austria, November 29 - December 2, 2011*, pages 272–281, 2011.

[8] Robert I. Davis and Alan Burns. A survey of hard real-time scheduling for multiprocessor systems. *ACM Comput. Surv.*, 43(4):35, 2011.

[9] Naveen Garg and Amit Kumar. Minimizing average flow-time : Upper and lower bounds. In *FOCS*, pages 603–613, 2007.

[10] Anupam Gupta, Sungjin Im, Ravishankar Krishnaswamy, Benjamin Moseley, and Kirk Pruhs. Scheduling heterogeneous processors isn't as easy as you think. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1242–1253, 2012.

[11] Dorit S. Hochbaum and David B. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM J. Comput.*, 17(3):539–551, 1988.

[12] Sungjin Im and Benjamin Moseley. An online scalable algorithm for minimizing ℓ_k-norms of weighted flow time on unrelated machines. In *ACM-SIAM Symposium on Discrete Algorithms*, 2011.

[13] R. Kumar, D.M. Tullsen, P. Ranganathan, N.P. Jouppi, and K.I. Farkas. Single-ISA heterogeneous multi-core architectures for multithreaded workload performance. In *Computer Architecture, 2004. Proceedings. 31st Annual International Symposium on*, pages 64 – 75, june 2004.

[14] Rakesh Kumar, Dean M. Tullsen, and Norman P. Jouppi. Core architecture optimization for heterogeneous chip multiprocessors. In *PACT '06: Proceedings of the 15th international conference on Parallel architectures and compilation techniques*, pages 23–32, New York, NY, USA, 2006. ACM.

[15] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, 1973.

[16] R. Merritt. CPU designers debate multi-core future. EE Times, Feb. 2010.

[17] Tomer Y. Morad, Uri C. Weiser, Avinoam Kolodny, Mateo Valero, and Eduard Ayguade. Performance, power efficiency and scalability of asymmetric cluster chip multiprocessors. *IEEE Comput. Archit. Lett.*, 5:4–, January 2006.