On Harmonic Fixed-Priority Scheduling of Periodic Real-Time Tasks with Constrained Deadlines[*]

Tianyi Wang[+] Qiushi Han[+] Shi Sha[+] Wujie Wen[+] Gang Quan[+] Meikang Qiu[†]
[+]Florida International University, Miami, FL
[†] Pace University, New York, NY
[+]{tiawang,qhan001,ssha001,wwen,gaquan}@fiu.edu, [†]mqiu@pace.edu

ABSTRACT

It is well known that a *harmonic* task set, i.e., task periods are integer multiples of each other, can better utilize a processor to achieve high system utilization. However, the current definition of *harmonic task set* is limited only to tasks with deadlines equal to their periods. In this paper, we extend the concept of "harmonic task set" to tasks with constrained deadlines, i.e., deadlines less than or equal to their periods. We show that a harmonic task set with constrained deadlines has a better schedulability than the non-harmonic one with the same task utilization. We employ this characteristic for task partitioning on multi-core platform, and our extensive experimental results show that, by taking the task harmonic relationship into consideration, our partitioning approach can greatly improve the schedulability of real-time tasks on multi-core platforms.

1. INTRODUCTION

Multi-core platforms have been mainstream for computing systems and more and more real-time computing systems will be built on multi-core platforms. Multi-core real-time scheduling can be generally categorized into three classes: the partitioned approach, the semi-partitioned approach, and the global approach. While different task partitioning approaches have different advantages/disadvantages and none of them really dominate the other in terms of schedulability [6], the partitioned approach—allocating each task to a dedicated processor—is usually adopted for real-time system design in practice for its better predicability and ease of design [1, 2].

When partitioning real-time tasks on multiple cores, one critical problem is how to partition real-time tasks such that processing cores can be most effectively utilized. This problem is a well known NP-hard problem [8] and designers usually have to resort to different heuristics with low computational cost. One such heuristic, for example, is simply to transform the problem into a bin-packing problem [7] and apply different bin-packing heuristics, such as first-fit (FF), worst-fit (WF) or best-fit (BF). In the general case, Andersson et al [1] proved that the utilization bound for multi-core partitioned approach with fixed-priority scheduling is only 50% per core.

It is a well-known fact that harmonic tasks [10], i.e., tasks with periods being integer multiples of each other, can achieve high utilization, i.e. as high as one, on a single processor according to the rate monotonic scheduling (RMS) policy. Kuo et al. [11] proposed to adjust loads on a single core processor by allocating harmonic tasks together. Han et al. [10] proposed a polynomial time method to determine the feasibility of a task set by verifying the feasibility of the corresponding harmonic task set transformed from the original task set. Bonifaci et al. presented exact feasibility checking algorithms with polynomial-time complexity for harmonic tasks under both fixed priority scheduling and dynamic priority scheduling [5]. Nasri et al. introduced a method to select harmonic periods for system utilization improvement [13]. Liu et al. [12] studied the problem of scheduling harmonic tasks with suspensions on both uniprocessor and multiple processors. Fan et al. proposed a multi-core partitioned scheduling algorithm that can take advantage of harmonic relationship among tasks to improve the system schedulability [9]. Wang et al. [14] explored the harmonic relationship among tasks with statistical execution times on multi-core platforms. All these works indicate that the system schedulability can be significantly improved if harmonic relationship among tasks can be exploited properly.

One great limitation of existing researches on harmonic real-time tasks is that they target solely on periodic tasks with implicit deadlines scheduled according to RMS scheme. Such an approach is not applicable for many practical real-time applications [5], which can be better modeled as periodic real-time tasks with constrained deadlines. In this paper, we extend the concept of "harmonic" from periodic tasks with *implicit* deadlines to the ones with *constrained* deadlines, scheduled according to deadline monotonic scheduling (DMS) policy. We formally define what it means for tasks with constrained deadlines to be harmonic. We show that, similar to a traditional harmonic task set, a *general* harmonic task set has a better schedulability than non-harmonic ones. Specifically, we formulate two theorems to demonstrate the high schedulability of a harmonic task set over a normal one. To our best knowledge, this is the first research effort that defines the "harmonic task set" for periodic tasks with constrained deadlines.

We then take task harmonic relationship into consideration to tackle the problem of partitioned fixed-priority scheduling of real-time tasks on homogenous multi-core platforms based on DMS scheme. Since not all tasks are perfectly harmonic, we develop a novel metric to quantify the degree

[*]This work is supported in part by NSF under projects CNS-1423137 and CNS-1457506.

DAC '16, June 05-09, 2016, Austin, TX, USA
© 2016 ACM. ISBN 978-1-4503-4236-0/16/06…$15.00
DOI: http://dx.doi.org/10.1145/2897937.2898055

of harmonicity between two tasks. Based on this metric, we then develop two partitioning algorithms that can take harmonic relationship of tasks into consideration. Extensive simulations results show that the proposed task partitioning approaches can significantly improve the schedulability of real-time tasks when compared with existing work.

2. PRELIMINARY

We consider a real-time system consisting of N independent periodic tasks, denoted as $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$, ordered by their priorities based on deadline monotonic scheduling (DMS) policy. Assume Γ is to be scheduled on a homogeneous multi-core platform, denoted as $P = \{p_1, p_2, ...p_M\}$, according to DMS. Each task $\tau_i \in \Gamma$ is characterized by a tuple (C_i, D_i, T_i), representing the worst case execution time, the relative deadline and the inter-arrival time (period), respectively. We assume $D_i \leq T_i$.

For each task $\tau_i = (C_i, D_i, T_i)$, we define its *density* (denoted by I_i) as $I_i = \frac{C_i}{D_i}$ and *utilization* (denoted by U_i) as $U_i = \frac{C_i}{T_i}$. Accordingly, the *utilization of a task set* Γ, denoted by U_Γ, is defined as $U_\Gamma = \sum_{i=1}^{N} U_i$. When task set Γ is scheduled on a multi-core system with K cores, we define the *system utilization* (denoted by U_s) as $U_s = \frac{U_\Gamma}{K}$. The problem of fixed-priority scheduling of periodic tasks with constrained deadlines on multi-core platforms can be formulated as follows:

Problem 1. *Given (*i*) task set $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$ and (*ii*) multi-core platform $P = \{p_1, p_2, ...p_M\}$, partition Γ on P such that all tasks can meet their deadlines and the number of cores used is minimized.*

A key to solve problem stated above is to partition real-time tasks in a way that can best utilize the processors. Consider the task set with six tasks shown in Table 1.

Table 1: A task set with six tasks

τ_i	C_i	D_i	T_i	U_i
τ_1	1	2	4	0.25
τ_2	1	3	4	0.25
τ_3	1	4	6	0.17
τ_4	1	5	6	0.17
τ_5	7	12	29	0.24
τ_6	7	12	38	0.18

One simple approach to partition the tasks above is to transform it into a traditional bin packing problem. Then we can apply heuristics such as FF, WF or BF to partition these tasks to different cores. Use FF as an example. First, we order these tasks according to the decreasing order of their utilizations, i.e., $\{\tau_1, \tau_2, \tau_5, \tau_6, \tau_3, \tau_4\}$. Then we allocate the tasks one by one to the first core that can accommodate the task. For the above example, we have τ_1, τ_2, τ_3 and τ_4 allocated to core 1, τ_5 and τ_6 allocated alone to core 2 and core 3, respectively. As such, to schedule tasks in Table 1 based on FF approach, at least three processing cores are needed.

Since harmonic tasks can better utilize a processor, an intuitive approach is therefore to allocate tasks with same periods (or tasks with periods being integer multiples of each other) to the same core. Specifically, for the six tasks above, we assign tasks τ_1 and τ_2 together to one processor and task τ_3, τ_4 and τ_5 to another processor. Again, since task τ_6 cannot be assigned to either of the two processors, we still have to allocate one more processor to schedule task τ_6. It is not difficult to verify that, if we assign task τ_1,τ_3 and τ_5 to one core and τ_2, τ_4 and τ_6 to another core, we can feasibly schedule all six tasks in two cores.

Note that both approaches above have their limitations. The first approach depends on the order of tasks to make partitioning decisions while the latter only takes harmonicity of task periods into consideration. Both approaches ignore the effects of deadline constraints for task partitioning. Also note that for a harmonic task set with implicit deadlines, the utilization can be as high as one. However, for task set with constrained deadlines, this is no longer always true any more, and partitioning tasks based on their periods becomes ineffective. We believe that, same as tasks with implicit deadlines, there must exist some *harmonic relationship* among tasks with constrained deadlines, and if this relationship is explored properly, we can greatly improve the processor utilization. The challenge is how to identify and quantify this relationship for periodic tasks with constrained deadlines. We discuss our approach for this problem in the sections that follow.

3. HARMONIC TASKS WITH CONSTRAINED DEADLINES

As indicated in the example above, for tasks with constrained deadlines, task sets with harmonic periods do not necessarily have better processor utilizations. The question is then what type of task sets may have a better utilization. The following example can shed some light on this question.

Table 2: A task set with three tasks.

τ_i	C_i	D_i	T_i
τ_1	1	2	3
τ_2	1	3	4
τ_3	C_3	D_3	24

Consider the task set shown in Table 2. Note that when we change τ_3's deadline D_3, its largest schedulable execution time $max(C_3)$ and the corresponding task density I_3 also change. Table 3 lists different values of D_3 and corresponding $max(C_3)$, I_3. For example, when we set task τ_3's deadline $D_3 = 8$, the corresponding largest execution time that can still make task τ_3 feasible is $max(C_3) = 3$ and the density $I_3 = \frac{C_3}{D_3} = \frac{3}{8} = 0.375$.

As shown in Table 3 and also illustrated in Figure 1, the density of task τ_3 changes non-monotonically with its deadline. It is interesting to note that task τ_3's density reaches its maximum when task τ_3's deadline equals to 12 and 24, or the integer multiples of task periods from τ_1 and τ_2. This seems to imply that when a lower priority task's deadline is an integer multiples of all higher priority tasks' periods, the lower priority task may achieve its maximum density without compromising its deadline. The higher the maximum density a task has, the more workload a processing core can accommodate. The task set therefore has a better schedulability. Based on this observation, we define the concept of *harmonic tasks* for periodic tasks with constrained deadlines as follows.

Table 3: Density changes for task τ_3.

$\mathbf{D_3}$	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
$max(\mathbf{C_3})$	1	2	2	3	3	3	4	5	5	5	6	6	6	7	7	8	8	8	9	10
$\mathbf{I_3}$	0.2	0.33	0.29	0.375	0.33	0.3	0.36	0.42	0.38	0.36	0.4	0.375	0.35	0.39	0.37	0.4	0.38	0.36	0.39	0.42

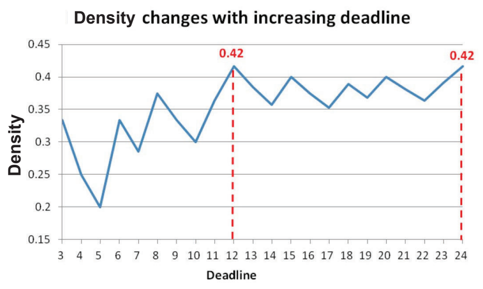


Figure 1: Density varies with different deadlines.

Definition 1. *Let τ_i and τ_j be two tasks with constrained deadline $D_i \le D_j$. Then τ_i and τ_j are* harmonic *if*

- *$T_i \le D_j$ and $T_i \mid D_j$ (i.e., T_i divides D_j), or*
- *$T_i > D_j$.*

In Definition 1, if the deadline of the low priority task is the integer multiple of the period of the high priority task then these two tasks are harmonic. This comes directly from the observation introduced above. On the other hand, if the high priority task's period is larger than the deadline of the low priority task, we also define two tasks being harmonic. This is because that, once the execution time of the high priority task is given, the density of the low priority task varies monotonically with its deadline, exhibiting the same behavior when the period of the high priority task equals to the deadline of the low priority task. Accordingly the harmonic task set with constrained deadlines is defined as follows.

Definition 2. *A task set is called a* general harmonic task set*, or simply* harmonic task set *if any two tasks in the task set are harmonic.*

From Definition 1 and Definition 2 we can see that the traditional harmonic tasks with implicit deadlines are just special cases of general harmonic tasks. While it is well known that for harmonic tasks with implicit deadlines the utilization bound is 1, this is not true any more for a general harmonic task set. To study the schedulability of a general harmonic task set, we have the following theorem (Proofs in this paper are omitted due to page limit.)

Theorem 1. *Let task set $\Gamma = \{\tau_1, \tau_2, \dots \tau_i, \dots, \tau_N\}$ be a harmonic task set with constrained deadlines. For $\tau_i \in \Gamma$, τ_i is schedulable* if and only if *the work demand of τ_i at the scheduling point $t = D_i$, i.e $W_i(D_i)$, is no more than D_i, where*

$$W_i(t) = C_i + \sum_{j=1}^{i-1} \lceil \frac{t}{T_j} \rceil C_j. \tag{1}$$

The sufficiency of this statement is readily true. To prove the necessity of this statement, we only need to note that there must be a scheduling point $t = kT_s \in (0, D_i], k \in Z$ such that $W_i(t) \le t$. Since $T_s \mid D_i$ and let $D_i = mT_s$ where $m \in Z$, we have $m > k$ and $D_i = \frac{m}{k}t$.

$$\begin{aligned} W_i(D_i) &= C_i + \sum_{\forall j<i} \lceil \frac{D_i}{T_j} \rceil \cdot C_j \\ &\le \frac{m}{k} C_i + \frac{m}{k} \sum_{\forall j<i} \lceil \frac{t}{T_j} \rceil \cdot C_j \\ &\le \frac{m}{k} t = D_i. \end{aligned}$$

From Theorem 1, we can see that, similar to traditional harmonic tasks, to check the schedulability of a task in a harmonic task set takes only linear time. More importantly, harmonic task sets defined by Definition 1 and 2 have better schedulability than that of non-harmonic ones. This conclusion is formulated in the following theorem.

Theorem 2. *Let $\Gamma = \{\tau_1, \tau_2, \dots \tau_i, \dots, \tau_N\}$ and $\Gamma' = \{\tau_1', \tau_2', \dots \tau_i', \dots, \tau_N'\}$ be two schedulable task sets with equal utilization, i.e., $U_\Gamma = U_{\Gamma'}$. Assume that Γ is a harmonic task set and Γ' is a regular task set. Let task $\tau_Z = (C_Z, D_Z, T_Z)$ be a task with priority lower than any task in Γ and Γ'. Then if $\{\Gamma' + \{\tau_Z\}\}$ is schedulable, then $\{\Gamma + \{\tau_Z\}\}$ must be schedulable.*

Theorem 2 indicates that for the same task τ_Z, if it is schedulable with a non-harmonic task set, it must be schedulable with a harmonic task set of the same utilization. Also, for a harmonic task set and a non-harmonic task set, if the corresponding tasks have the same utilizations and densities, then if the non-harmonic task set is schedulable, the harmonic task set must be schedulable, as formally formulated in the following theorem.

Theorem 3. *Let $\Gamma = \{\tau_1, \tau_2, \dots \tau_i, \dots, \tau_N\}$ and $\Gamma' = \{\tau_1', \tau_2', \dots \tau_i', \dots, \tau_N'\}$ be two task sets, and let $U_i = U_i'$ and $I_i = I_i'$ for any $\tau_i \in \Gamma$ and $\tau_i \in \Gamma'$. Assume Γ is harmonic. Then if Γ' is schedulable, Γ must be schedulable.*

It is not difficult to see that Theorem 3 can be applicable for traditional harmonic tasks with implicit deadlines. That is, if a task set is schedulable, then a harmonic task set with the same utilization must be schedulable. If tasks have constrained deadlines, however, we have to take the deadline constraints into consideration and require their densities are equal.

Now let us revisit the motivation example. As mentioned before, the task sets can be scheduled using two processors: $\{\tau_1, \tau_3, \tau_5\}$ and $\{\tau_2, \tau_4, \tau_6\}$. If we pay a close attention to the first subset $\{\tau_1, \tau_3, \tau_5\}$, we can see that τ_3's deadline is an integer multiple of τ_1's period, and task τ_5's deadline is integer multiples of periods for both τ_1 and τ_3. Therefore this partition helps to reduce the number of processors. Similar observation can be made from the other subset. Note that τ_4's deadline is very close to an integer multiple of τ_2's period, and task τ_6's deadline is integer multiples of periods for both τ_2 and τ_4.

As the motivation example implies, if we take the harmonic relationship into consideration, we may significantly improve the processor utilization. The problem, however, is that not all tasks in a task set are perfectly harmonic. How can we quantify which tasks are more harmonic than others? In what follows, we first introduce a metric to quantify the degree of harmonicity between two tasks. Based on this metric, we then propose two algorithms to guide our partition procedure on multi-core platforms.

4. THE HARMONIC INDEX

In this section, we introduce the metric that we have developed to quantify the degree of harmonicity between two tasks. Specifically, the *harmonicity* of two tasks is measured by the "distance" of a task to the harmonic task. Before we introduce the metric in details, we first introduce the following definitions.

DEFINITION 3. *Given two tasks $\tau_i = (C_i, D_i, T_i)$ and $\tau_j = (C_j, D_j, T_j)$ with $D_i \le T_i \le D_j$, the* harmonic sub-task *of τ_j with respect to τ_i is task $\tau'_j = (C_j, D'_j, T_j)$, such that D'_j is the largest value with $D'_j \le D_j$ and $T_i \mid D'_j$. On the other hand, the* harmonic sub-task *of τ_i with respect to τ_j is task $\tau'_i = (C_i, D_i, T'_i)$, such that T'_i is the largest value with $T'_i \le T_i$ and $T'_i \mid D_j$.*

In other word, for the *low* priority task, its harmonic sub-task is the task with the exactly the same execution time and period, but the largest possible *deadline* (not larger than the original one) that is harmonic with the high priority task. On the other hand, for the *high* priority task, its harmonic sub-task is the task with exactly the same execution time and deadline, but the largest possible period (not larger than the original one) harmonic to the low priority task. In this way, when replacing the original task with its harmonic sub-task and the result task set is schedulable, the original task set must be schedulable. We formulate this conclusion in the following theorem.

THEOREM 4. *Let $\Gamma = \{\tau_1, \tau_2, \ldots \tau_i, \ldots, \tau_N\}$ and let τ'_i be a harmonic sub set with respect to any task $\tau_j \in \Gamma$. Then if task set $\{\tau_1, \tau_2, \ldots \tau'_i, \ldots, \tau_N\}$ is feasible, Γ must be feasible.*

Now we are ready to formally define harmonic index to evaluate how a task is harmonic to the other.

DEFINITION 4. *Given two tasks τ_i and τ_j with $D_i \le T_i \le D_j$, let τ'_j (τ'_i, resp) be the harmonic sub task of τ_j (τ_j, resp) with respect of τ_i (τ_j, resp). Then the* harmonic index *of τ_j (τ_i, resp) with respect of τ_i (τ_j, resp), denoted as $\mathcal{H}(\tau_j \to \tau_i)$ ($\mathcal{H}(\tau_i \to \tau_j)$, resp), is defined as*

$$\mathcal{H}(\tau_j \to \tau_i) = I'_j - I_j, \quad (2)$$
$$\mathcal{H}(\tau_i \to \tau_j) = U'_i - U_i. \quad (3)$$

where I'_j and I_j are densities of τ'_j and τ_j, respectively, and U'_i and U_i are utilizations of τ'_i and τ_i, respectively. The harmonic index *of these two tasks, denoted as $\mathcal{H}(\tau_j, \tau_i)$ is defined as*

$$\mathcal{H}(\tau_i, \tau_j) = min\{\mathcal{H}(\tau_j \to \tau_i), \mathcal{H}(\tau_i \to \tau_j)\}. \quad (4)$$

The metrics of $\mathcal{H}(\tau_j \to \tau_i)$ and $\mathcal{H}(\tau_i \to \tau_j)$ define how close a task is to its corresponding harmonic sub task in terms of density/utilization change. The larger the change is, the less harmonic the task is to the reference task. Therefore a high harmonic index value indicates a low harmonic relationship.

So far, we discuss the case when the high priority task's period is no larger than low priority task's deadline, i.e., $D_i \le T_i \le D_j$. If the high priority task's period is greater than the low priority task's deadline, we consider the two tasks are harmonic. That is,

$$\mathcal{H}(\tau_j \to \tau_i) = \mathcal{H}(\tau_i \to \tau_j) = \mathcal{H}(\tau_i, \tau_j) = 0. \quad (5)$$

The rationale behind this definition is that, when high priority task has a period longer than the deadline of the low priority task, the high priority task preempts the low priority task only one time, which is the exactly the same when the high priority task has the period equal to the deadline of the low priority task.

With the definition of harmonic index for two tasks, we can also extend it to the entire task set as follows.

DEFINITION 5. *Given a task set $\Gamma = \{\tau_1, \tau_2, ..., \tau_i, ..., \tau_N\}$, the harmonic index of task τ_i in task set Γ, denoted as $\mathcal{H}_\Gamma(\tau_i)$, is defined as:*

$$\mathcal{H}_\Gamma(\tau_i) = \sum_{\tau_j \in \Gamma} \mathcal{H}(\tau_i, \tau_j) \quad (6)$$

5. TASK PARTITIONING ALGORITHMS

With the harmonic indexes we have defined in the previous section, we are now ready to introduce our task partitioning algorithms. The goal of our task partitioning algorithms is to identify tasks that have high harmonicity and group them into one processor to better utilize resources. To this end, we propose two algorithms. The first algorithm, called *greedy density maximization algorithm* (GIM), allocates one task at a time to the core with task set that is most harmonic to the task. The second algorithm, called *harmonic-aware clique maximization algorithm* (HCM), addresses the task partition problem from a higher perspective. It first identifies tasks that are harmonic or close to harmonic and then assign them to a processing core together.

5.1 Greedy density maximization algorithm

The greedy density maximization algorithm (GIM) is based on the harmonic index which we have proposed earlier. The details of the algorithm is shown in Algorithm 1. Given a task set and a multi-core platform, the algorithm first sorts tasks in non-increasing order of their utilizations (line 4). Then it goes through each task and allocates each task to the best candidate processor that is schedulable and most harmonic to this task.

Algorithm 1 is a simple yet effective approach and the timing complexity is only $O(N^2(M + max(D)))$, where N is the total number of tasks, M is the number of cores, and $max(D)$ is the pseudo polynomial complexity for response time analysis of harmonic task sets. Since Algorithm 1 allocates one task at a time, the order of the tasks can significantly affect the partitioning choice and a task can only be grouped with existing tasks that have assigned to a core.

5.2 Harmonic-aware clique maximization algorithm

The second algorithm, harmonic-aware clique maximization algorithm (HCM), intends to identify tasks that have high harmonicity first and then allocate them to a core. Different from GIM, tasks assigned to a core may potentially

Algorithm 1 Greedy density maximization algorithm.

```
1: Input:$\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$ and $P = \{p_1, p_2, \ldots, p_M\}$;
2: Output: Task partitions: $= \{\Gamma_{p_1}, \Gamma_{p_2}, \ldots, \Gamma_{p_M}\}$.
3:
4: Sort tasks in $\Gamma$ by the non-increasing order of their utilizations;
5: for each task $\tau_i \in \Gamma$ do
6:   for each processor $p_j \in P$ do
7:     Calculate $\mathcal{H}_{\Gamma_{p_j}}(\tau_i)$ if allocating $\tau_i$ to $p_j$;
8:     Check the feasibility of $\Gamma_{p_j}$ if allocating $\tau_i$ to $p_j$;
9:   end for
10:  Allocate $\tau_i$ to $p_j$ such that $\Gamma_{p_j}$ is feasible and $\mathcal{H}_{\Gamma_{p_j}}(\tau_i)$ is the minimum;
11: end for
```

have a higher harmonicity and therefore can achieve better performance.

To identify tasks with high harmonicity, one intuitive approach (similar to [9]) is to rank the harmonic indexes for all tasks based on each candidate task and then pick the ones with smaller harmonic indexes. However, different from the harmonic tasks with implicit deadlines, the general harmonic relationship is not transitive. That is: if task A and B are harmonic and task B and C are harmonic, it does not necessary mean that task A and C are harmonic. To search for sub task sets allocated to a processor, HCM transforms this problem to the classic maximum clique problem [4], which intends to find the largest fully connected subgraph in a graph. When different edges have different weights, the maximum clique problem becomes the one to identify the sub-graph with maximized/minimized total weights. In HCM, we let each task be a node in the graph and the harmonic index be the weight for edge connecting two nodes. Then the clique with the minimum total weight corresponds to the task set with the minimum harmonic index.

The maximum clique problem is a NP-hard problem in nature and many heuristics have been proposed, such as greedy algorithm, simulated annealing, neural network and etc [4] where the timing complexity is a serious concern. In HCM, we apply a greedy heuristic to address this problem with a timing complexity of O(N^4), where N is the total number of tasks. The details of the algorithm is shown in Algorithm 2.

HCM searches for a feasible clique in each iteration and returns the one with maximum utilization. In each iteration the algorithm constructs cliques with tasks that are harmonic or close to harmonic. Tasks are added to a clique until no further tasks can be added with all tasks being schedulable (If two tasks have the same harmonic index, the one with larger density is picked). Then HCM sorts all candidate cliques in non-increasing order of their utilizations and picks the clique with the maximum utilization to allocate to a core. The tasks in the clique are then removed from task set Γ. This process is repeated until task set Γ is empty.

6. EXPERIMENTAL RESULTS

In this section, we use experiments to investigate the effectiveness of our proposed algorithms. Four different approaches were realized in our experiment: one traditional bin packing approach (FF), i.e., the first fit decreasing, the harmonic approach that does not consider deadline constraints

Algorithm 2 Harmonic-aware clique maximization algorithm.

```
1: Input:$\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$ and $P = \{p_1, p_2, \ldots, p_M\}$;
2: Output: Task partitions: $= \{\Gamma_{p_1}, \Gamma_{p_2}, \ldots, \Gamma_{p_M}\}$.
3: $k = 1$;
4: while $\Gamma \neq \emptyset$ do
5:   Calculate $\mathcal{H}(\tau_i, \tau_j)$ for each $\tau_i, \tau_j \in \Gamma$;
6:   Search for the schedulable $\Gamma_{p_k} \subseteq \Gamma$ with the largest cliques [4];
7:   $\Gamma = \Gamma - \Gamma_{p_k}$;
8:   $k = k + 1$;
9: end while
```

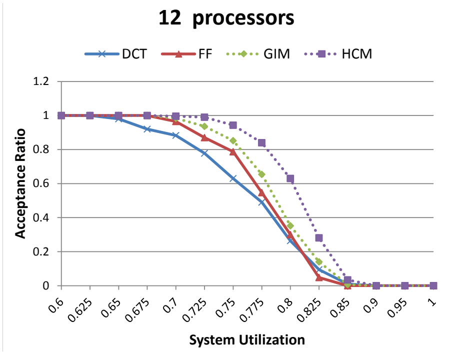


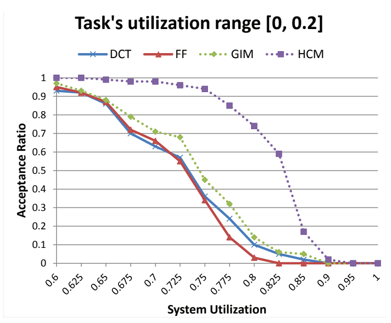
Figure 2: Performance vs. system utilizations

(DCT) [9], and two task partitioning algorithms introduced above, i.e. *GIM* and *HCM*.

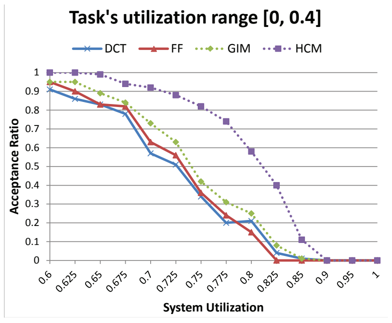
We studied the performance of different approaches with respect to different system utilizations. We randomly generated task sets with total system utilization from 0.6 to 1 with step size of 0.025. Task periods are evenly distributed within interval [500, 1000] and deadline-period ratios are evenly distributed between [0.2, 1]. We modified the UUniFast approach [3] such that the maximum utilization for a task is no more than $U_{max} = 0.2$. We set the number of cores to be 12. For each setting, we ran over 1000 experiments and calculated the average acceptance ratio (i.e. schedulable task sets vs. total number of task sets) for different approaches. The results are shown in Figure 2.

From Figure 2, we can see that both *GIM* and *HCM* can always achieve better performance than *DCT* and *FF*. *DCT* is the worst approach among the four since *DCT* determines if tasks are harmonic based only on periods, which does not work well if task deadlines are much smaller than their periods. From our results, we can see that even *FF* tends to perform better than *DCT*. It is interesting to note that *GIM* outperforms *FF* with a relatively low timing complexity (due to cost for feasibility checking). *HCM* is the best approach due to the fact that it searches the harmonic tasks from the entire task set for a core and therefore can exploit the harmonic relationship more effectively. Specifically, when system utilization is 0.8, the acceptance ratio by *HCM* almost doubles the ones by *DCT* and *FF*.

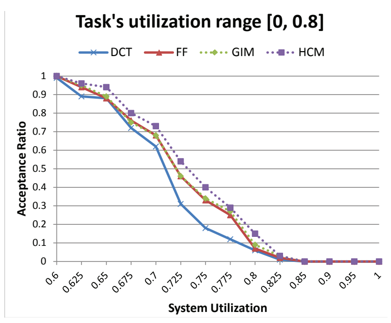
We next studied the performance of each approach with respect to different numbers of tasks. We varied the maximum utilization for each task U_{max} from 0.2 to 0.4, and then to 0.8. Note that the larger that U_{max} is, the smaller number of tasks are there in a task set. Again, we ran experiments for each setting 1000 times and average acceptance ratio was calculated and shown in Figure 3.



(a)



(b)



(c)

Figure 3: Performance vs. number of tasks on a 12-core processor

From the experimental results shown in Figure 3, we can see that when task's maximum utilization increases, the performance of each algorithm decreases. This is because large maximum utilization implies smaller number of tasks in the task set, and thus less flexibility for each algorithm to choose tasks. We can also see that HCM always outperforms other approaches in all test cases, and the improvement increases with the number of tasks, which demonstrates that HCM can better explore the harmonic relationship among tasks to better schedule tasks on multi-core platforms. For example, in Figure 3, for system with total utilization of 0.8, HCM improves upon GIM, DCT and FF over 107% when $U_{max} = 0.8$ (Figure 3(d)), and over 523% when $U_{max} = 0.2$ (Figure 3(d)). The results clearly show that by taking advantage of harmonic relationship among task, our approaches can significantly improve the multi-core scheduling performance.

7. CONCLUSIONS

For fixed-priority multi-core scheduling problem, one key problem is how to partition tasks in a way that can most effectively utilize the resource. While there have been extensive approaches that exploit harmonic relationship among tasks to improve system utilization, they are limited only to real-time tasks with implicit deadlines. In this paper, we extend the concept of "harmonic task set" to tasks with constrained deadlines and show that a *general* harmonic task set with constrained deadlines always has a better schedulability than the non-harmonic one. We employ this characteristic for task partitioning on multi-core, and extensive simulation results show that our algorithms can significantly outperform existing approaches. As far as we know, this is the first research that defines the "harmonic task set" for periodic tasks with constrained deadlines. We believe that this research can greatly benefit many existing researches that exploit harmonic relationship of real-time tasks.

8. REFERENCES

[1] B. Andersson and J. Jonsson. The utilization bounds of partitioned and pfair static-priority scheduling on multiprocessors are 50%. In *Euromicro Conference on Real-Time Systems*, pages 33–33. IEEE Computer Society, 2003.

[2] E. Bini and G. C. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Syst.*, 30(1-2):129–154, May 2005.

[3] E. Bini and G. C. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Systems*, 30(1-2):129–154, 2005.

[4] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo. The maximum clique problem. In *Handbook of combinatorial optimization*, pages 1–74. Springer, 1999.

[5] V. Bonifaci, A. Marchetti-Spaccamela, N. Megow, and A. Wiese. Polynomial-time exact schedulability tests for harmonic real-time tasks. In *Real-Time Systems Symposium*, pages 236–245. IEEE, 2013.

[6] J. Carpenter, S. Funk, P. Holman, A. Srinivasan, J. Anderson, and S. Baruah. A categorization of real-time multiprocessor scheduling problems and algorithms. *Handbook on Scheduling Algorithms, Methods, and Models, pages*, pages 30–1, 2004.

[7] E. G. Coffman Jr, M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: a survey. In *Approximation algorithms for NP-hard problems*, pages 46–93. PWS Publishing Co., 1996.

[8] R. I. Davis and A. Burns. A survey of hard real-time scheduling for multiprocessor systems. *ACM Computing Surveys (CSUR)*, 43(4):35, 2011.

[9] M. Fan and G. Quan. Harmonic-aware multi-core scheduling for fixed-priority real-time systems. *Parallel and Distributed Systems, IEEE Transactions on*, 25(6):1476–1488, June 2014.

[10] C.-C. Han and H. ying Tyan. A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithms. In *Real-Time Systems Symposium*, pages 36–45, Dec 1997.

[11] T.-W. Kuo and A. K. Mok. Load adjustment in adaptive real-time systems. In *Real-Time Systems Symposium*, pages 160–170. IEEE, 1991.

[12] C. Liu, J. J. Chen, L. He, and Y. Gu. Analysis techniques for supporting harmonic real-time tasks with suspensions. In *26th Euromicro Conference on Real-Time Systems*, pages 201–210, July 2014.

[13] M. Nasri, G. Fohler, and M. Kargahi. A framework to construct customized harmonic periods for real-time systems. In *26th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 211–220. IEEE, 2014.

[14] T. Wang, L. Niu, S. Ren, and G. Quan. Multi-core fixed-priority scheduling of real-time tasks with statistical deadline guarantee. In *Design, Automation & Test in Europe Conference & Exhibition*, pages 1335–1340. EDA Consortium, 2015.