Scheduling periodic tasks in a hard real-time environment ⋆

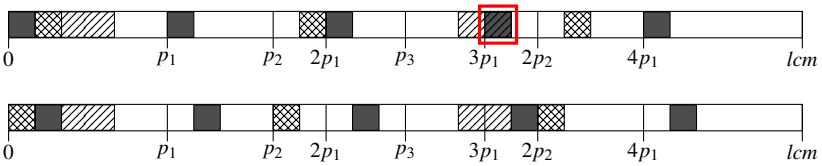
Friedrich Eisenbrand[1], Nicolai Hähnle[1], Martin Niemeier[1],
Martin Skutella[2], José Verschae[2], and Andreas Wiese[2]

[1] EPFL, Lausanne, Switzerland
[2] TU Berlin, Germany

Abstract We give a rigorous account on the complexity landscape of an important real-time scheduling problem that occurs in the design of software-based aircraft control. The goal is to distribute tasks $\tau_i = (c_i, p_i)$ on a minimum number of identical machines and to compute offsets a_i for the tasks such that no collision occurs. A task τ_i releases a job of running time c_i at each time $a_i + k \cdot p_i$, $k \in \mathbb{N}_0$ and a collision occurs if two jobs are simultaneously active on the same machine. Our main results are as follows: (i) We show that the minimization problem cannot be approximated within a factor of $n^{1-\varepsilon}$ for any $\varepsilon > 0$. (ii) If the periods are harmonic (for each i, j one has $p_i \mid p_j$ or $p_j \mid p_i$), then there exists a 2-approximation for the minimization problem and this result is tight, even asymptotically. (iii) We provide asymptotic approximation schemes in the harmonic case if the number of different periods is constant.

1 Introduction

The motivation for this research comes from a real-word combinatorial optimization problem that was communicated to us by our industrial partner, a major avionics company. The aircraft designers need to schedule highly critical periodic control tasks with a predictable and static scheduling policy such that preemption and dynamic effects are avoided. The model that is used in this context is as follows. One is given *tasks* $\tau_1, \ldots, \tau_n$ where each task $\tau_i = (c_i, p_i)$ is characterized by its *execution time* $c_i \in \mathbb{N}$ and *period* $p_i \in \mathbb{N}$. The goal is to assign the tasks to identical machines and to compute offsets $a_i \in \mathbb{N}_0$ such that no collision occurs. A task τ_i generates one *job* with execution time c_i at every time unit $a_i + p_i \cdot k$ for all $k \in \mathbb{N}_0$. Each job needs to be processed immediately and non-preemptively after its generation on the task's machine. A collision occurs if two jobs are simultaneously active on one machine.



The picture above shows three tasks $\tau_1 = (1,6), \tau_2 = (1,10)$ and $\tau_3 = (2,15)$. The upper part shows an infeasible assignment of offsets ($a_1 = 0,\ a_2 = 1,\ a_3 = 2$) whereas

⋆ This work was partially supported by Berlin Mathematical School, by DFG research center MATHEON, by the DFG Focus Program 1307 within the project "Algorithm Engineering for Real-time Scheduling and Routing", and by the Swiss National Science Foundation.

the lower part shows a feasible assignment of offsets ($a_1 = 1,\ a_2 = 0,\ a_3 = 2$). Notice that the schedule repeats after the least-common multiple (*lcm*) of the periods.

In the single machine context and with unit execution times, this problem was studied by Wei and Liu [WL83] who called the problem of computing offsets the *periodic maintenance problem*. Baruah et al. [BRTV90] and independently [BBNS02,Bha98] show that the periodic maintenance problem is NP-hard in the strong sense.

The engineers are however interested in the corresponding *machine minimization* problem, i.e., they want to find the minimum number of identical machines on which the tasks can be distributed in a feasible way. We refer to this problem as the *periodic maintenance minimization problem*. Korst et al. [KALW91,KAL96] studied this problem and show independently from [BRTV90] and [BBNS02,Bha98] that it is NP-hard in the strong sense. It occurs in particular when additional features have to be implemented on a given architecture. For the avionics company it is then preferable to re-design the control software only rather than to re-design and change the hardware components. Sometimes even moderately sized real-word instances turn out to be unsolvable with state-of-the-art integer programming approaches. One feature that the easier instances share is that, with only few exceptions, their tasks have harmonic periods, i.e., for each pair of tasks τ_i, τ_j one has $p_i \mid p_j$ or $p_j \mid p_i$. Thus, one question that arises is whether instances with this divisibility property are easier to solve than general instances.

For many combinatorial optimization problems, the answer to an analogous question is indeed *yes*. For instance, KNAPSACK with divisible items [Mar85,VA97] or the MIXING SET problem with divisible capacities [ZIRdF08,CSW08] are only two examples, where one has polynomial time algorithms on the one hand for the divisible case and NP-hardness for the general case. For the periodic maintenance minimization problem that is in the focus of this work the difference is however drastic and reflects very well the experience with those real-word instances whose tasks mostly have harmonic periods and very general instances.

Our contribution is a rigorous account on the complexity and approximability landscape of the above described machine-minimization problem. Our results are summarized in Table 1. In this extended abstract, we limit ourselves to the description of the following results that are highlighted in this table. More results and details are presented in the full version of this paper [EHN$^+$10].

► We prove that, for any $\varepsilon > 0$, it is NP-hard to approximate the periodic maintenance minimization problem within a factor of $n^{1-\varepsilon}$, i.e., that the trivial approximation algorithm is essentially tight. This explains the difficulty of moderately sized instances without harmonic periods from a theoretical viewpoint. The result is achieved by a reduction from COLORING that relies on basic number-theoretic results like the Chinese Remainder Theorem and the Prime Number Theorem. We remark that the reduction has been given independently in [BBNS02,Bha98]. The hardness result also holds under resource augmentation.

► We show that the periodic maintenance minimization problem with harmonic periods allows for a 2-approximation algorithm. Furthermore, we show that this is tight, even asymptotically. It is remarkable that a simple variant of First-Fit can be analyzed to be a 2-approximation algorithm. The analysis differs however considerably from the simple analysis that shows that First-Fit for BIN-PACKING yields a 2-approximation.

arbitrary periods		
period lengths k	algorithms	hardness results
k arbitrary	$2OPT + k - 1$	$n^{1-\varepsilon} OPT$
k constant	$\left(\frac{3}{2} + \varepsilon\right) OPT + k$	$\left(\frac{3}{2} - \varepsilon\right) OPT + k - 1$

harmonic periods		
period lengths k	algorithms	hardness results
k arbitrary	$2OPT$	$(2-\varepsilon)\, OPT + o(OPT)$
k constant	$(1+\varepsilon)\, OPT + k$	$\left(\frac{3}{2} - \varepsilon\right) OPT + k - 1$
q_k/q_1 constant	$(1+\varepsilon)\, OPT + 1$	$(2-\varepsilon)\, OPT$

Table 1. The approximability landscape of the periodic maintenance minimization problem. Here q_k and q_1 denote the largest and smallest period length, respectively.

The novel concept that we use is the one of a witness for certain groups of machines. These witnesses prove that these groups are heavily loaded. If a witness for a certain group of machines is missing, the instance can be separated into two independent sub-instances. This allows for an analysis by induction.

► Even though the 2-approximation result for the case of harmonic periods is tight, we show that a stronger restriction leads to an asymptotic PTAS: If the number of different periods k is constant, we have an efficient algorithm with approximation guarantee $(1+\varepsilon)OPT + k$, for any constant $\varepsilon > 0$. The basic approach follows the ideas of the classical APTAS for BIN-PACKING of Fernandez de la Vega and Luecker [FdlVL81]. The more complicated nature of the periodic maintenance problem, however, requires several interesting and nontrivial extensions of the techniques such as a more sophisticated rounding procedure and advanced structural insights into the solutions to the periodic maintenance problem. This helps to enumerate the solution space to find a template that can be turned into a solution with the desired approximation guarantee.

Related work. There is excessive literature on real-time scheduling; for surveys see, e.g., [BHR93,But04,Leu04]. In contrast to our problem, one is often interested in verifying whether a set of tasks can be scheduled on one machine with a certain scheduling policy. The periodic maintenance minimization problem generalizes BIN-PACKING. In fact, if the periods of tasks are all identical, the problem is equivalent to BIN-PACKING. The problem is, however, more general and the approximation ratios known for BIN-PACKING do not carry over. While there is, for instance, a 1.5-approximation algorithm for BIN-PACKING [SL94], achieving this performance ratio is impossible for our problem, even for the case of harmonic periods.

2 Harmonic periods

First-Fit is a simple and very popular heuristic for many packing problems, such as, e.g., BIN-PACKING etc. Adapted to our problem, First-Fit considers tasks in some given order and greedily packs the current task on the first open machine on which it fits. In case there is no such machine, it opens a new machine on which the current task is scheduled.

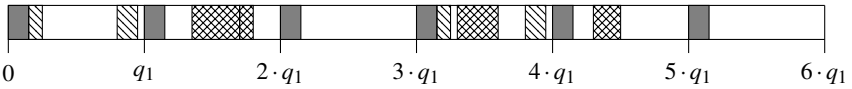


Figure 1. A schedule for a single machine. The gray jobs belong to tasks with period length q_1, the striped jobs to tasks with period length $q_2 = 3 \cdot q_1$, and the checkered jobs to tasks with period length $q_3 = 6 \cdot q_1$.

A crucial subproblem occurring in this context is to decide whether a task τ can be scheduled on a machine on which other tasks, say $\tau_1, \ldots, \tau_n$, have already been scheduled without changing the offsets of these tasks. If all processing times are equal to 1 and the period of τ is p, we will see in Section 3 that the feasible offsets a for task τ are the solutions to the system

$$a \not\equiv a_j \bmod \gcd(p, p_j) \quad \text{for } j = 1, \ldots, n. \tag{1}$$

For arbitrary (not harmonic) $p_j \in \mathbb{N}$ and $p = \prod_{i=1}^{n} p_j$, this is the NP-complete problem of computing *simultaneous incongruences*, see, e.g., [GJ79]. Thus, for instances of the periodic maintenance minimization problem with arbitrary periods, already this crucial subproblem is NP-hard and it is not clear how to implement First-Fit.

In this section we consider the special case of harmonic periods. In Section 2.1 we show how the crucial subproblem can be solved efficiently in this case by making clever use of a special solution structure.

In the following we always assume that tasks $\tau_1 = (c_1, p_1), \ldots, \tau_n = (c_n, p_n)$ are sorted such that $p_j \mid p_{j+1}$, for $j = 1, \ldots, n-1$. Moreover, the number of different period lengths of the input-tasks is denoted by k and the set of all task periods is denoted by $Q = \{q_1, \ldots, q_k\}$, where $q_i \mid q_{i+1}$, for $i = 1, \ldots, k-1$.

2.1 Bin trees

An important encoding of single-machine schedules which we use repeatedly in this paper is the (compressed) *bin tree* that we now explain. Assume that offsets a_j for tasks $\tau_j = (c_j, p_j)$, $j = 1, \ldots, n$, are given. Then the resulting *schedule*, i.e., which task runs a job at which time, repeats itself after the largest period q_k. The smallest period q_1 partitions the time-horizon $[0, q_k)$ into *bins* $[i \cdot q_1, (i+1) \cdot q_1)$; see Figure 1 for an example.

Assume that the offsets a_j, $j = 1, \ldots, n$, determine a feasible single-machine schedule for tasks $\tau_1, \ldots, \tau_n$ where no two jobs collide. By a simple shifting argument, we can assume w.l.o.g. that the first task τ_1 with minimum period $p_1 = q_1$ has offset $a_1 = 0$ (see also Figure 1). Now consider an additional task $\tau = (c, p)$ with maximum period $p = q_k$. Clearly, one can find a feasible offset for this task if and only if one of the q_k/q_1 bins has a free slot of size c.

Consider two bins $B_i = [i \cdot q_1, (i+1) \cdot q_1)$ and $B_j = [j \cdot q_1, (j+1) \cdot q_1)$ such that $i \equiv j \mod q_r/q_1$. As far as tasks with period length up to q_r are concerned, these bins look the same. As a shorthand, we write $B_i \equiv_r B_j$ when $i \equiv j \mod q_r/q_1$.

The root of a *full bin tree* is a node representing a bin containing all tasks of period length q_1. It has q_2/q_1 children, each of which represents a bin that contains all its parent's tasks and may contain additional tasks of period length q_2. We say that the root is of period q_1, and its children are of period q_2.

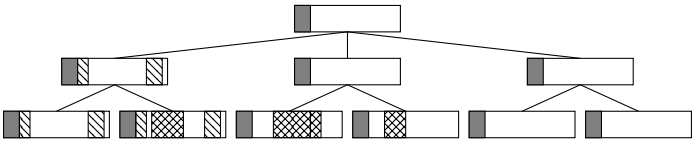


Figure 2. The full bin tree corresponding to the schedule in Figure 1.

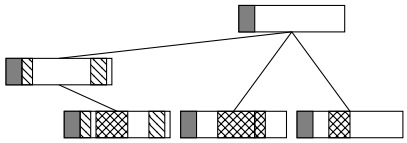


Figure 3. The compact form of the bin tree in Figure 2.

In general, a node B of period q_r contains only tasks of period length up to q_r. If $q_r < q_k$, it has q_{r+1}/q_r children, each of which is a node of period q_{r+1}. Each child of B represents a bin that contains all tasks of B and may contain additional tasks of period length q_{r+1}. Each scheduled task of period length q_r appears in a unique node of period q_r and in all children of that node.

As a consequence of this definition, there is a one-to-one correspondence between nodes of period q_r in the full bin tree and equivalence classes of bins modulo the equivalence relation $\equiv_r$. Furthermore, the hierarchy of equivalence relations $\equiv_r$ ($r = 1, \ldots, k$) corresponds to the hierarchy of the tree in the following way: If two nodes of period $\geq q_r$ have the same ancestor of period q_r, then their corresponding bins are equivalent modulo $\equiv_r$, and vice versa. In particular the leafs of the bin-tree correspond to the bins of the schedule. Thus we can freely convert between a feasible schedule in terms of task offsets and the corresponding bin tree representation; see Figure 2.

The number of nodes in a full bin tree is dominated by its leaves, of which there are q_k/q_1 many, so we cannot operate efficiently on full bin trees and, in particular, we cannot afford to store a full bin tree to implement our First-Fit heuristic. However, if a node of period q_r does not contain a task of period q_r, it is completely determined by its parent. Therefore, we only need to store those nodes of the tree that introduce a new task to the schedule, see Figure 3 for an example. In this way we have a *compressed bin tree* whose number of nodes is bounded by the number of tasks and that can be constructed in polynomial time.

Coming back to the implementation of First-Fit, given a bin tree and an unscheduled task τ whose period length is greater or equal to the period lengths of all tasks in the bin tree, one can easily determine whether τ can be inserted into the bin tree in compact form by checking all the leaves of the compact tree, as well as all nodes to which a new leaf of the right period can be added. We will use this fact in the next subsection.

2.2 First-Fit algorithm

In the sequel we use the following term: If a subset of tasks is assigned to a machine, then the *type* of this machine is the smallest period of these tasks. The First-Fit algorithm maintains a list $M_1, \ldots, M_\ell$ of open machines where M_i was opened before M_j if $i < j$.

The algorithm is initialized with the empty list. Then, the algorithm proceeds as follows. For $j = 1, \ldots, n$:

1. Find the first machine on which τ_j fits and insert it into a leaf of that machine's bin tree such that no gaps are created within the leaf. Within this machine's bin tree, break ties between leaves arbitrarily.
2. If τ_j does not fit on any open machine, we open a new machine of type p_j and add τ_j to the root node of its bin tree. Furthermore, to simplify the analysis later, we open a *second* new machine of type p_j. On this machine we schedule a dummy task with running time 0 and period p_j.

Note that, because tasks are added in non-decreasing order of period lengths, we only add tasks to leaves of the compressed bin tree as discussed above.

The *utilization* of a task $\tau = (c, p)$ is defined as $\text{util}(\tau) = c/p$. For a set of tasks I and a machine M we define $\text{util}(I) = \sum_{\tau \in I} \text{util}(\tau)$ and $\text{util}(M) = \sum_{\tau \text{ on } M} \text{util}(\tau)$, respectively. The utilization $\text{util}(I)$ is a lower bound for $OPT(I)$. The main result of this section is the following theorem.

Theorem 1. *The First-Fit algorithm is a* 2*-approximation algorithm.*

Before we present the proof of Theorem 1 we will discuss some differences to the analysis of the First-Fit algorithm for BIN-PACKING and motivate the concept of a *witness* that turns out to be very useful. The simple observation showing that First-Fit for BIN-PACKING is a 2-approximation is as follows: If First-Fit opens a new bin, then let α be the minimum weight of all previously opened bins. This implies that the current item has weight at least $1 - \alpha$ and if there are any other open bins, they must have weight at least $\max\{\alpha, 1 - \alpha\}$. The average weight of the bins is thus at least $1/2$.

Now suppose that the First-Fit algorithm for the machine minimization problem opens a new machine for task $\tau_j = (c_j, p_j)$. If the *type* q of some machine with low utilization is smaller than the running time c_j of the task, then this task cannot be run on this machine. Thus, it may happen that there are many open machines with a low utilization. In particular, it is not true that the average utilization of the open machines is at least $1/2$. However, we can derive a lower bound on the average load of the machines whose type is *compatible* with τ_j, where the set of compatible types is denoted by $Q(j) := \{q \in Q \colon c_j \leq q \leq p_j\}$.

Lemma 1. *Suppose that the First-Fit algorithm cannot schedule τ_j on one of its open machines and opens two new machines instead. Let $q \in Q(j)$ be a compatible machine type, and let $M_1, \ldots, M_\ell$, $\ell > 0$, be the machines of type q that were open before the algorithm tries to assign τ_j. Then, $\frac{1}{\ell} \sum_{i=1}^{\ell} \text{util}(M_i) > \frac{1}{2}$.*

Proof. First observe that $\ell \geq 2$ because the First-Fit algorithm always opens two machines of the same type at a time. Consider the bin trees corresponding to machines $M_1, \ldots, M_\ell$ when τ_j should be added. Note that the leaves of the trees are of period at most p_j. Let $\alpha > 0$ be the minimum fill ratio over all leaf-bins of the trees. If $\alpha > \frac{1}{2}$, then every bin is more than half filled and the claim follows.

Thus, we can assume that $\alpha \leq \frac{1}{2}$. Let $\overline{B}$ be a leaf bin of fill ratio α, and $\overline{M}$ be the machine $\overline{B}$ belongs to. Thus, the utilization of $\overline{M}$ is at least α. We will show that all

leaf bins of the machines $\{M_1,\dots,M_\ell\}\setminus\{\overline{M}\}$ have a fill ratio greater than $1-\alpha$. This implies, in particular, that all machines other than $\overline{M}$ have a utilization greater than $1-\alpha$. Since $\ell \geq 2$, the claim then follows by an averaging argument.

Let B be a leaf bin on a machine $M_i \neq \overline{M}$. There are two cases to consider: The First-Fit algorithm considers M_i *before* or *after* $\overline{M}$. If M_i is considered before $\overline{M}$, let τ be any task assigned to $\overline{B}$. Now consider the time when τ was assigned by First-Fit. At that time, either B or an ancestor of B was a leaf-bin. First-Fit tried to assign τ to B (or its ancestor) but failed. Now τ fills at most an α-fraction of a bin, which implies that the fill ratio of B (or its ancestor) must have been more than $1-\alpha$, otherwise τ would have been packed there instead. Thus, B has fill ratio greater than $1-\alpha$ at the time τ is assigned. For the case where M_i is considered after $\overline{M}$, we can analogously argue that a task in B could have been assigned to $\overline{B}$ (or an ancestor). □

Let τ_j be a task and let $\mathscr{M}_j$ be the set of machines that are open when First-Fit tries to assign τ_j. Let $Q' \subseteq Q(j)$ be a subset of the compatible machine types and let $\mathscr{M}' \subseteq \mathscr{M}_j$ be the subset of machines whose type is in Q'. The above lemma implies that, if First-Fit opens two new machines when it tries to assign τ_j, then the average load of the machines in $\mathscr{M}'$ is at least $1/2$. We say that τ_j is a *witness* of $\mathscr{M}'$. In particular, if $q < p_j$ is compatible with τ_j and if $\mathscr{M}_q$ denotes the machines of type q that are created by First-Fit, then τ_j is a witness of $\mathscr{M}_q$. We have the following lemma.

Lemma 2. *$FF(I)$ denotes the number of machines opened by the First-Fit heuristic. If for all $q \in Q$, $q < q_k$, the set $\mathscr{M}_q$ has a witness, then $FF(I) \leq 2OPT(I)$.*

Proof. Let $q \in Q$ with $q < q_k$, and let τ be a witness for $\mathscr{M}_q$. Then we can apply Lemma 1 to show that $\sum_{M\in\mathscr{M}_q} \text{util}(M) \geq \frac{1}{2}|\mathscr{M}_q|$. Now let $\mathscr{M}'$ be the set $\mathscr{M}_{q_k}$ without the two last machines that we call $\widetilde{M}_1$ and $\widetilde{M}_2$. Let τ be a task assigned to $\widetilde{M}_1$. Observe that τ is a witness for $\mathscr{M}'$. Thus, $\sum_{M\in\mathscr{M}'} \text{util}(M) \geq \frac{1}{2}|\mathscr{M}'|$. Hence we have $FF(I)-2 \leq 2\cdot\text{util}(I\setminus\{\tau\})$ which implies $FF(I)-2 < 2\cdot\text{util}(I) \leq 2\cdot OPT(I)$. Since both $FF(I)$ and $2\cdot OPT(I)$ are even, we conclude $FF(I) \leq 2\cdot OPT(I)$. □

If the special case of Lemma 2 does not apply, we can identify sub-instances that are pairwise independent of each other, yet cover all machines opened by First-Fit. We use this observation to prove Theorem 1 by induction.

Proof (of Theorem 1). We prove the theorem by induction on k, the number of different periods. If $k = 1$, then the claim is obvious. Now assume that First-Fit is a 2-approximation for all instances with less than k periods. If for all $q \in Q$, $q < q_k$, the set $\mathscr{M}_q$ has a witness, again the claim follows directly with Lemma 2.

Thus, let $q \in Q$, $q < q_k$ be a period such that $\mathscr{M}_q$ does not have a witness. We now partition the tasks into $I' := \{\tau \in I\colon \text{ First-Fit assigns } \tau \text{ to a machine of type } \leq q\}$ and $I'' := I\setminus I'$. Moreover, let $\bar{I} := \{\tau_i \in I'\colon p_i \leq q\}$. Let τ_j be an arbitrary task in I''. Then q is not compatible with τ_j since otherwise τ_j would be a witness for $\mathscr{M}_q$. Thus, each task in I'' has a running time $> q$. As the period lengths of all tasks in $\bar{I}$ are at most q, no task in $\bar{I}$ can be scheduled together with a task in I''. This shows that $\bar{I}$ and I'' are independent in the sense that $OPT(\bar{I}) + OPT(I'') = OPT(\bar{I}\cup I'') \leq OPT(I)$.

On the other hand, since First-Fit assigns each task of I' to a machine of type $\leq q$ and these must have been opened and typed by a job in $\bar{I}$, one has $FF(\bar{I}) = FF(I')$ and $FF(I) = FF(\bar{I}) + FF(I'')$. Using the induction hypothesis, we get

$$FF(I) = FF(\bar{I}) + FF(I'') \leq 2OPT(\bar{I}) + 2OPT(I'') \leq 2OPT(I).$$

This concludes the proof. □

Surprisingly, the approximation result in Theorem 1 is tight, even if one aims for asymptotic approximation guarantees. This is in sharp contrast to the classical BIN-PACKING problem, where one has a fully polynomial asymptotic approximation scheme [KK82] and for which one does not know variants of First-Fit with optimal (asymptotic) approximation ratios. The proof of the following theorem can be found in [EHN+10]. It is established via a reduction from PARTITION and boosting.

Theorem 2. *Unless $P = NP$, there is no approximation algorithm with a guarantee of $(2-\varepsilon)OPT + o(OPT)$ for the case of harmonic periods, for any $\varepsilon > 0$.*

2.3 An APTAS for a constant number of periods

For the case of harmonic periods the 2-approximation algorithm cannot be improved, unless $P = NP$. This holds even asymptotically. In particular, there is no hope for an asymptotic PTAS. Our experience with real-world instances from our industrial partner, however, is that these instances often have only very few different period lengths. We thus consider the case where the number of different periods k is bounded by a constant and present an asymptotic PTAS for this restricted setting.

Theorem 3. *For any $\varepsilon > 0$ and k bounded by a constant, there is an efficient algorithm that computes a solution of value at most $(1+\varepsilon)OPT(I) + k$.*

We only sketch the main ideas of the algorithm. For a complete description we refer to the full version of this paper [EHN+10]. Although the high level idea is the same as for the APTAS for BIN-PACKING [FdlVL81], the more complex nature of the periodic maintenance problem imposes novel and interesting difficulties. In particular, we cannot simply classify tasks as *big* or *small*, since different bin sizes occur on different machines. Therefore we must do this classification relative to the size of the bin, making the rounding procedure significantly more involved. After the rounding, we determine the optimal solution for the big tasks by enumeration, making non-trivial use of the concept of bin trees. Finally, we add the small tasks using First-Fit. All these steps have to be performed with extreme care not to introduce extra additive factors to the guarantee on the objective function.

First, we show that the individual schedule for each machine can be assumed to have a particular structure. More precisely, we say that a schedule obeys the *bin structure* if each bin satisfies the following (see also Figure 4): (i) Let q be the smallest period length of a task on the machine; then, there is a task τ_j with $p_j = q$ and $a_j = 0$; (ii) if a task τ_j starts before a task $\tau_{j'}$ in the bin, then $p_j \leq p_{j'}$; and (iii) all jobs in a bin are processed consecutively.

Notice that the same bin tree structure described in Section 2.1 can be used for solutions that follow the bin structure. Moreover, a simple swapping argument shows the following lemma.

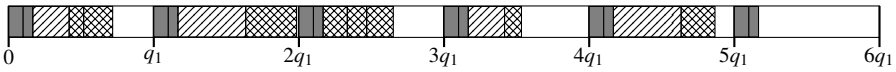


Figure 4. A schedule in bin structure. The gray jobs belong to tasks with period length q_1, the striped jobs to tasks with period length $q_2 = 3 \cdot q_1$ and the checkered jobs to tasks with period length $q_3 = 6 \cdot q_1$.

Lemma 3. *Let I' be a set of tasks assigned to a machine M. If there is a feasible schedule for I', then there is a feasible schedule that obeys the bin structure.*

The structure given by this lemma is crucial to describe schedules compactly: Given an assignment of tasks to bins in the bin tree such that no bin is overloaded, we can find an offset for each task such that no two tasks collide. In particular, in the enumeration step of our APTAS, we will only enumerate over the assignments of the tasks to the bins, not the individual schedules within each bin.

Rounding. In order to round the execution times of big tasks, more forethought is necessary compared to the APTAS for BIN-PACKING. We want to achieve two goals: On the one hand, only a constant number of big tasks should fit into a bin. On the other hand, we want to ensure that, later on, the remaining space in the bins can be used efficiently by the First-Fit procedure for the small tasks.

Given a bin size $q \in Q$, we say that a task τ_j is *small* with respect to q if $c_j < \varepsilon q$ and *big* if $\varepsilon q \leq c_j \leq q$. Now the rounding is done independently for each period length q_ℓ: For $i = 1, \ldots, k$, in iteration i, we only round tasks that are big with respect to q_i, and that have not been rounded before. These tasks are sorted by non-decreasing execution times and then partitioned into $O(1/\varepsilon^2)$ many groups. All groups are of equal size except for the first group which may have fewer elements. The execution times of all tasks of the same group are then rounded up to the maximum execution time of that group.

Denote by I' the rounded instance. For each period length q_ℓ and each bin size q_i, denote by $L_{\ell,i}$ the set of tasks on the last group. We define $L := \cup_{\ell,i} L_{\ell,i}$ and $J := I' \setminus L$. Using $OPT(I)$ as a template for J, one can show that $OPT(J) \leq OPT(I)$. For each bin size q, denote by C_q the set of all execution times c of big tasks in J such that $c \leq q$ (and thus a task with execution time c could possibly be scheduled in such a bin). The rounding ensures that the cardinality of each set C_q is bounded by a constant.

We will show at the end that the tasks in L together with a certain subset of the small tasks can be packed onto $O(\varepsilon)OPT(I) + k$ extra machines. Now, we proceed with presenting an approximation algorithm for J.

Enumeration of Solutions. In order to enumerate the solutions of big tasks in J, it is crucial to compactly represent the bin trees. We introduce the concept of a *bin configuration* which, given a bin schedule, encodes: (a) the level of a bin on the bin tree; (b) the size of the bin q; (c) the assignment of big tasks; and (d) the space reserved for small tasks for each period length, rounded down to the nearest multiple of εq.

Since the number of big tasks in a bin is at most $1/\varepsilon$, and $|C_q|$ is bounded by a constant, the total number of bin configurations is also bounded by a constant. We can therefore enumerate the bin configurations on the compressed bin trees in polynomial time as follows:

(i) Assume that, for each machine, we have already determined the bin configurations of all bins up to level q_i in its corresponding compressed bin trees.
(ii) For each bin configuration of level q_{i+1}, we enumerate the number of bins following this configuration. In what follows, we consider only the iteration in which we enumerate the respective amount that corresponds to the optimal solution to J.
(iii) Attach each node configuration to a parent node on any machine, such that the jobs of period lengths up to q_i coincide.

One of the enumerated solutions corresponds—up to permutation of sub-trees—to an optimal solution to J. Indeed, by an inductive argument, we can assume that levels up to q_i contains the right number of nodes of each configuration. Then, if in Step (ii) we enumerate the correct number of bins with each configuration, it is clear that Step (iii) will successfully attach the nodes to the trees.

Moreover, the running time of the algorithm is polynomial. Indeed, since the number of configurations on each level is at most a constant λ, Step (ii) can be performed in time $O(n^{\lambda})$. Thus, the running time of the whole algorithm is bounded by $O(n^{k\lambda})$.

The APTAS. We are now ready to state the APTAS: (1) Round tasks and define instances J and L; (2) enumerate over all possible configurations of bin trees; (3) assign the big tasks in J according to the bin configurations; (4) greedily assign small tasks to bins by using the reserved space in the configurations; call S the set of small tasks that could not be assigned; (5) for each period length q independently, schedule all jobs in $L \cup S$ of period q with the classical First-Fit algorithm for BIN-PACKING. (Notice that, in this last step, we do not mix tasks of different period lengths on the same machine.)

Up to Step (4) of the algorithm, we have only used $OPT(J) \leq OPT(I)$ many machines. It remains to show that First-Fit in Step (5) assigns tasks in $L \cup S$ to at most $O(\varepsilon)OPT + k$ machines. We omit the proof and refer to [EHN$^+$10].

Lemma 4. *First-Fit assigns tasks in $L \cup S$ to at most $O(\varepsilon)OPT + k$ machines.*

If not only k is bounded by a constant but even q_k/q_1, one can obtain a better bound.

Theorem 4. *For any $\varepsilon > 0$ and q_k/q_1 bounded by a constant, there is an efficient algorithm that computes a solution of value at most $(1+\varepsilon)OPT(I) + 1$.*

The main difference to the algorithm above is that here a task τ_i is small if $c_i \leq \varepsilon q_1$ and big otherwise. Also, we need to be more careful with the First-Fit procedure to assign the remaining small jobs as in Step (5). For full details we refer to [EHN$^+$10].

3 Arbitrary period lengths

In this section we study the periodic maintenance problem in the setting of arbitrary, i.e., not necessarily harmonic, period lengths. We only sketch the results. For more details see [EHN$^+$10].

One can derive a First-Fit algorithm as follows: Partition the tasks according to their period lengths. Then do First-Fit for period lengths separately such that no two jobs with different period lengths are scheduled on the same machine. Denote by $FF(I)$ the number of machines in the resulting schedule. Using utilization bound techniques similar to BIN-PACKING, we obtain the following theorem.

Theorem 5. *For any instance I it holds that $FF(I) \leq 2 \cdot OPT + k - 1$.*

Now assume that k is bounded by some constant and let $\varepsilon > 0$. We call a task τ *small* if $\text{util}(\tau) \leq \varepsilon$, otherwise we call τ *big*. We define the sets I_{small} and I_{big} respectively. We enumerate all solutions for the big tasks and call the best solution $ENUM(I_{big})$. This can be done in polynomial time since there are at most $\lfloor 1/\varepsilon \rfloor$ big tasks on each machine and k is assumed to be constant (a similar argument as used in [FdlVL81] for BIN-PACKING). We output $EFF(I) := \min\{FF(I), ENUM(I_{big}) + FF(I_{small})\}$.

Theorem 6. *Assume that k is bounded by a constant. Then, $EFF(I)$ can be computed in polynomial time and $EFF(I) \leq (3/2 + O(\varepsilon))\, OPT + k$.*

In the remainder of this section we show that the periodic maintenance minimization problem is hard to approximate within a factor of $n^{1-\varepsilon}$ using an approximation preserving reduction from COLORING. We remark that this reduction has been stated independently in [BBNS02,Bha98]. Again, we only sketch the results and refer to [EHN$^+$10] for more details. The result still holds if we restrict the problem to unit execution times.

For unit execution times, a set of offsets is feasible if and only if $a_i + k_i \cdot p_i \neq a_j + k_j \cdot p_j$ for all i, j and $k_i, k_j \in \mathbb{N}_0$. With elementary number theory [NZM91] it is easy to see that this is equivalent to $a_i \not\equiv a_j \bmod \gcd(p_i, p_j)$. The reduction works as follows. Let the graph $G = (V, E)$ be an instance of the COLORING problem. Let $\overline{E}$ be the complement of E, i.e., $\overline{E} := \{\{u, v\} : u, v \in V, \{u, v\} \notin E\}$. We choose pairwise different primes q_e for all $e \in \overline{E}$. This can be done in polynomial time with the sieve of Eratosthenes since the Prime Number Theorem guarantees $\Theta(x/\ln(x))$ primes among the first x natural numbers (see, e.g., [NZM91]). For each node $v \in V$, we define a task $\tau_v = (c_v, p_v)$ with $c_v := 1$ and $p_v := \prod_{e \in \overline{E}:\, v \in e} q_e$. We denote with $\text{red}(G) := \{\tau_v : v \in V\}$ the periodic maintenance instance obtained from the graph G using this construction. Note that we can compute $\text{red}(G)$ in polynomial time. Our construction has the following properties:

Lemma 5. *Given a graph $G = (V, E)$, the machine minimization instance* $\text{red}(G)$ *satisfies the following properties.*

a) For each edge $\{u, v\} \in E$, tasks τ_u and τ_v cannot be assigned to the same machine.
b) For an independent set $U \subseteq V$, tasks $\{\tau_v : v \in U\}$ can be scheduled on one machine.

These properties can be used to show that our construction is a reduction.

Lemma 6. *For each $k \in \mathbb{N}$, a graph G is k-colorable if and only if* $\text{red}(G)$ *can be scheduled on k machines.*

It is shown in [Zuc07] that it is NP-hard to approximate the COLORING problem within a factor of $n^{1-\varepsilon}$, for any constant $\varepsilon > 0$, where n is the number of nodes of the graph. This immediately implies the claimed inapproximability result.

Theorem 7. *The periodic maintenance minimization problem cannot be approximated within a factor of $n^{1-\varepsilon}$, for any constant $\varepsilon > 0$, unless $P = NP$.*

We remark that the primes generated in the reduction can get exponentially large (although their encoding length is polynomial). For that reason, this result only shows that it is weakly NP-hard to approximate within the factor given in Theorem 7. The question whether there is a pseudopolynomial time algorithm with a constant factor approximation guarantee remains open.

References

[BBNS02] Amotz Bar-Noy, Randeep Bhatia, Joseph Naor, and Baruch Schieber. Minimizing service and operation costs of periodic scheduling. *Math. Oper. Res.*, 27(3), 2002.

[Bha98] R. Bhatia. *Approximation Algorithms for Scheduling Problems*. PhD thesis, University of Maryland, 1998.

[BHR93] S. K. Baruah, R. R. Howell, and L. E. Rosier. Feasibility problems for recurring tasks on one processor. In *Selected papers of the 15th International Symposium on Mathematical Foundations of Computer Science*, pages 3–20. Elsevier, 1993.

[BRTV90] S. Baruah, L. Rousier, I. Tulchinsky, and D. Varvel. The complexity of periodic maintenance. *Proceedings of the International Computer Symposium*, 1990.

[But04] G. C. Buttazzo. *Hard Real-time Computing Systems: Predictable Scheduling Algorithms And Applications*. Springer, 2004.

[CSW08] M. Conforti, M. Di Summa, and L. A. Wolsey. The mixing set with divisible capacities. In *Proceedings of the 13th International Conference on Integer Programming and Combinatorial Optimization (IPCO 2008)*, pages 435–449, 2008.

[EHN+10] F. Eisenbrand, N. Hähnle, M. Niemeier, M. Skutella, J. Verschae, and A. Wiese. Scheduling periodic tasks in a hard real-time environment. Technical report, EPF Lausanne & TU Berlin, February 2010. Available for download at http://disopt.epfl.ch/webdav/site/disopt/shared/PM_EHNSVW10_report.pdf.

[FdlVL81] W. Fernandez de la Vega and G. S. Lueker. Bin packing can be solved within $1+\varepsilon$ in linear time. *Combinatorica*, 1:349–355, 1981.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. Freemann, 1979.

[KAL96] J. Korst, E. Aarts, and J. K. Lenstra. Scheduling periodic tasks. *INFORMS Journal on Computing*, 8:428–435, 1996.

[KALW91] J. Korst, E. Aarts, J. K. Lenstra, and J. Wessels. Periodic multiprocessor scheduling. In E. H. L. Aaarts, J. van Leeuwen, and M. Rem, editors, *PARLE '91 Parallel Architectures and Languages Europe*, volume 505 of *Lecture Notes in Computer Science*, pages 166–178. 1991.

[KK82] N. Karmakar and R. M. Karp. An efficient approximation scheme for the one-dimensional binpacking problem. In *Foundations of Computer Science, (FOCS 23)*, pages 312–320, 1982.

[Leu04] J. Y.-T. Leung. *Handbook of Scheduling: Algorithms, Models and Performance Analysis*. Chapman & Hall/CRC, 2004.

[Mar85] O. Marcotte. The cutting stock problem and integer rounding. *Mathematical Programming*, 33:82–92, 1985.

[NZM91] I. Niven, H. S. Zuckerman, and H. L. Montgomery. *An Introduction to the Theory of Numbers, 5th edition*. Wiley, 1991.

[SL94] D. Simchi-Levi. New worst-case results for the bin-packing problem. *Naval Research Logistics*, 41:579–585, 1994.

[VA97] W. F. J. Verhaegh and E. H. L. Aarts. A polynomial-time algorithm for knapsack with divisible item sizes. *Information Processing Letters*, 62:217–221, 1997.

[WL83] W. D. Wei and C. L. Liu. On a periodic maintenance problem. *Operations Research Letters*, 2:90–93, 1983.

[ZIRdF08] M. Zhao and Jr. I. R. de Farias. The mixing-MIR set with divisible capacities. *Mathematical Programming*, 115:73–103, 2008.

[Zuc07] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3:103–128, 2007.