Schedulability Analysis of Real-Time Digraph Tasks Scheduled with Static Priority

Chao Peng · Haibo Zeng

Received: date / Accepted: date

Abstract Graph-based task models have been studied to better model and analyze the schedulability of real-time systems. Among them, the digraph task model, with its powerful expressiveness to describe the behavior of a large class of real-time tasks, receives a wide range of interests recently. However, the exact schedulability analysis of digraph tasks scheduled with static priorities is shown to be coNP-hard. Approximate analyses based on the request and interference bound functions (r bf and i bf) are proposed to improve the analysis efficiency. In this work, we summarize the existing results on these analysis techniques, and seek to further improve their generality, complexity, and accuracy. Specifically, we develop analysis techniques for tasks with arbitrary deadlines. We prove the periodicity of interference bound function such that it can be expressed as a finite aperiodic part and an infinite periodic part, which makes the asymptotic complexity of its calculation independent from the length of the time interval. Moreover, we develop a linear upper bound on i bf that is tighter than that of r bf, to derive a better response time bound.

Keywords Digraph task models · Schedulability analysis · Request/Interference bound functions

1 Introduction

Real-time systems typically consist of a number of concurrent tasks sharing the limited hardware resources. To perform the schedulability analysis, system designers
need to construct workload models for tasks to abstract their requirements on computing resources. A series of graph based workload models with different expressiveness and analysis complexity is proposed, such that the verification of the time constraints can be studied and analyzed formally. In these graph based models, vertices represent different kinds of jobs, and edges determine the possible flows of control. Each vertex (job) is characterized by its worst case execution time (WCET) and relative deadline. Each graph edge is labeled with the minimum separation time between the releases of the two vertices (jobs) it connects. Please see a recent survey by Stigge and Yi [22].

Among these graph-based models, Liu and Layland introduce the first and simplest, the Periodic Task Model [14]. Each periodic task releases job instances based on a constant period and these jobs have the same WCETs. Mok proposes the Sporadic Task Model to allow a new job to be released anytime after a constant time has passed since the release of its previous job [16]. The Multiframe Task Model [17] and Generalized Multiframe Task Model (GMF) [3] both assume that the WCETs are not constant but defined according to a cyclic pattern, hence with a linear release structure. The Recurring Branching Task Model provides a further relaxation by allowing selection points to determine the behaviors of the released jobs, which can be modeled as a directed tree [4]. The Recurring Real-Time Task Model allows task graph to be any directed acyclic graph [5]. The Digraph Task Model (DRT) removes the restriction of recurrence by allowing arbitrary cycles and therefore arbitrary directed graphs, which can represent arbitrary release structure of jobs and significantly increase the expressiveness [20]. In this paper, we consider the digraph real-time task model [20], which is a strict generalization of the previous models [3–5, 14, 16, 17]. The digraph model is shown to be expressive enough to describe the real-time workload in synchronous finite state machines [27], such as those modeled in popular tools like Stateflow [1] and SynchCharts [2, 8].

Several more expressive models are also proposed. The Extended Digraph Task Model (EDRT) is proposed in [21] by introducing global inter-release separation constraints to DRT models. The Synchronous Digraph Real-Time Task Model (SDRT) [15] extend the DRT model to allow inter-task synchronization through a rendezvous mechanism. More specifically, each edge in the digraph may be augmented with a synchronous action. Sink jobs of edges that are from different tasks but share the same synchronous action shall be released at the same time. The model of timed automata with tasks [18] allows complex dependencies between job release times and task synchronization. The relationships among these models are summarized in a number of papers such as [20, 29].

The schedulability analysis of digraph task models is extensively studied. For systems with dynamic priority, Earliest Deadline First is known to be optimal, and the analysis is shown to be tractable, i.e., no worse than pseudo-polynomial time for bounded utilization systems [20]. However, for systems with static priority, the complexity is much higher. The exact schedulability analysis for GMF tasks is already coNP-hard in the strong sense: the complemented problem to check if the task set is unschedulable is NP-hard [23]. It is proved by the reduction from 3-PARTITION, a well-known NP-hard problem [9]. Essentially, the exact analysis requires to check the schedulability using the total execution requests for each of the path combinations from the higher priority tasks. To improve the efficiency, Stigge and Yi introduce a
method called combinatorial abstraction refinement, by iteratively refining the abstraction of paths until the schedulability condition is satisfied [24, 25]. However, the worst case complexity remains to be the same. Also, the result is limited to systems with frame separation property (job deadline is no larger than the release time of the next job).

A more efficient analysis method is to settle with sufficient only schedulability conditions. Guan et al. propose an approximate response time analysis by using the request bound function ($rbf$), the maximum execution request for any path in the digraph [11]. It also derived a more accurate analysis leveraging the interference bound function ($ibf$), the maximum workload of a task exclusively executed on the CPU. Both of these two approximate analysis methods, based on $rbf$ and $ibf$ respectively, have pseudo-polynomial time complexity. Also, they both provide bounded speed-up factors [11]. To improve the efficiency of calculating the request and demand bound functions for DRT tasks, Zeng and Di Natale leverage results from max-plus algebra to demonstrate the linear periodicity of the two functions, which makes the complexity asymptotically independent from the length of the time interval [28, 29]. In this work, we prove that the interference bound functions are also linear periodic.

Bini et al. [7] present a linear upper bound on the worst case response time for sporadic tasks. It uses the concept of load executed at higher priority, which essentially is the same as the total $ibf$ from higher priority tasks. For DRT tasks, two linear upper bounds on the response time, with different degrees of tightness, are proposed in [20] and [29] respectively. In this paper, we also derive a closed-form upper bound on the worst case response time by developing a linear upper bound on the interference bound function and study its approximation quality.

1.1 Contributions and Paper Organization

We make a few contributions to further generalize and improve schedulability analysis techniques for digraph real-time task model. We provide analysis for tasks with arbitrary deadlines. We prove the linear periodicity of the interference bound function, i.e., it can be represented by a finite aperiodic part and a periodic part repeated infinitely often. We also develop a linear upper bound on $ibf$. We study the effectiveness of the approach by demonstrating that it provides a tighter upper bound on the response time than existing approaches [20, 29].

The rest of the paper is organized as follows. Section 2 describes the digraph task model. Section 3 summarizes the existing work on the workload functions and schedulability analysis. It also demonstrates that the analysis based on either ($rbf$) or ($ibf$) has no bounded approximation ratio, and derives exact and approximate response time analysis techniques for tasks with arbitrary deadlines. Section 4 proves the linear periodicity of interference bound functions by task transformation. Section 5 presents the linear upper bound of the interference bound function, studies its speedup factor, and applies it to get a linear upper bound on the response time. Section 6 shows the experimental evaluation on the improved analysis efficiency, as well as the quality of the linear response time upper bound. Finally, Section 7 concludes the paper.
2 Digraph Task Model

A digraph real-time task (DRT) $\tau$ is characterized by a directed graph $D(\tau) = (\mathbb{V}, \mathbb{E})$ where the set of vertices $\mathbb{V} = \{v_1, v_2, \ldots\}$ represents the types of jobs that can be released for task $\tau$. For convenience, we use $D(\tau)$ to refer to the DRT as well. Each vertex $v_i \in \mathbb{V}$ (or type of job) is characterized by an ordered pair $\langle e(v_i), d(v_i) \rangle$, where $e(v_i)$ and $d(v_i)$ denote its worst case execution time (WCET) and relative deadline, respectively. Edges represent possible flows of control, i.e., the release order of the jobs of $\tau$. An edge $(v_i, v_j) \in \mathbb{E}$ is labeled with a parameter $p(v_i, v_j)$ that denotes the minimum separation time between the releases of $v_i$ and $v_j$. We assume that all the task parameters (WCET, relative deadline, and minimum inter-release time) are positive integers (i.e., in $\mathbb{N}^+$). For simplicity, we also use the following notation

$$
\begin{align*}
    k_{i,j} &= p(v_i, v_j), \quad k_i = \max\{1, \max_{v_j: (v_i, v_j) \in \mathbb{E}} k_{i,j}\}, \quad m_i = e(v_i)
\end{align*}
$$

As a simple condition for schedulability, we assume that the WCET of any node is no larger than the minimum inter-release time of any of its outgoing edges

$$
\forall (v_i, v_j) \in \mathbb{E}, \quad e(v_i) \leq p(v_i, v_j)
$$

We consider three types of deadline settings, discussed in increasing generality below:

- A task has the **frame separation property** [5] (also defined as constrained deadlines [11]) if its jobs satisfy

$$
\forall (v_i, v_j) \in \mathbb{E}, \quad d(v_i) \leq p(v_i, v_j)
$$

That is, the deadline of any job is no larger than the minimum inter-release time of its outgoing edges, which guarantees that any job finishes before the release of the next job if the system is schedulable.

- A more relaxed condition is the **1-MAD property** [5] if

$$
\forall (v_i, v_j) \in \mathbb{E}, \quad d(v_i) \leq d(v_j) + p(v_i, v_j)
$$

In this case, the absolute deadline of a job can be larger than the release time, but not the absolute deadline, of the sink job for any of its outgoing edges. This property is more relaxed than the frame separation property, but it is still sufficient to guarantee that the absolute deadline of the last job is the largest among all jobs.

- If a task has no restriction on the inter-release times and the deadlines, we say this task has **arbitrary deadlines**, under which the deadline of any job can take any value.

**Example 1** Figure 1 shows an example of a digraph real-time task with 3 vertices (types of jobs). Node $v_1$ is associated with a pair $(2, 3)$, which indicates that $v_1$ has a WCET of 2 and a relative deadline of 3. Edge $(v_1, v_2)$ has a weight of 4, meaning the release times of $v_1$ and $v_2$ are separated by at least 4 time units. In the task graph of Figure 1, all the edges satisfy Equation (2), thus the task satisfies the frame separation property.
A task system consists of a set of independent real-time tasks \{\tau_1, \tau_2, \ldots\}. We assume that tasks are scheduled on a uni-processor with preemptive static-priority scheduling. The task set is defined to be schedulable, if and only if all job sequences generated by the tasks in the set can be executed such that all jobs meet their deadlines. The complexity of the exact analysis is proven to be coNP-hard [23].

3 Schedulability Analysis with Workload Functions

3.1 Workload Functions

We first summarize the workload abstraction functions that are useful for the analysis, organized by their abstraction level. Path-level workload abstraction defines one function for each path in the task graph, while task-level abstraction provides one for each task.

3.1.1 Path-level functions

The exact analysis involves the calculation of the workload at the path level [24, 25]. We first give a few relevant definitions.

Definition 1 In a digraph task \(D(\tau) = (V, E)\), a path \(\pi\) is a sequence of nodes \((v_1, v_2, \ldots, v_l)\) where each \((v_k, v_{k+1})\) is an edge in \(E\), which models some specific execution sequence of the task. If there exist no repetitive nodes in the path, \(\pi\) is called an elementary path. For a path \(\pi\), the length \(|\pi|\) of \(\pi\) is the sum of \(p(v_k, v_{k+1})\) of each edge \((v_k, v_{k+1})\) in the path. The weight \(e(\pi)\) is the sum of the WCET \(e(v_i)\) of each node \(v_i\) in the path. The mean \(\lambda(\pi)\) of \(\pi\) is defined as

\[
\lambda(\pi) = \frac{e(\pi) - e(v_1)}{|\pi|} = \frac{\sum_{i=1}^{l-1} e(v_i)}{\sum_{i=1}^{l-1} p(v_i, v_{i+1})}
\]

If \(v_1 = v_1\), \(\pi\) is called a cycle. Similarly, if there exist no repetitive nodes in the cycle except \(v_1\) and \(v_l\), \(\pi\) is called an elementary cycle. The maximum cycle mean of \(D(\tau)\), i.e., the maximum among the cycle means in \(D(\tau)\), is denoted as \(\lambda(\tau)\), which is the same as the task utilization [20, 29]. We further define the maximum path

Fig. 1 An example of digraph real-time task.
mean in \( D(\tau) \) as \( \lambda^*(\tau) \). Obviously \( \lambda^*(\tau) \geq \lambda(\tau) \). A digraph is strongly connected if there exists a path between any pair of vertices.

Stigge et al. [20] proposes an approach to calculate \( \lambda(\tau) \), which has pseudo-polynomial time complexity. This approach is based on the observation that it is sufficient to consider only the elementary cycles to calculate \( \lambda(\tau) \) (Lemma V.5 in the paper). Similarly, we can also calculate \( \lambda^*(\tau) \) by enumerating all the elementary paths, as stated in the following lemma. It is a trivial extension of Lemma V.5 in [20], as the proof also applies to paths (not only cycles), and we omit the proof here.

**Lemma 1** For any path \( \pi \), there is an elementary path \( \pi' \) such that \( \lambda(\pi') \geq \lambda(\pi) \).

The following two functions, the request function \( \text{rf} \) and interference function \( \text{if} \), capture the execution request and executed load respectively for a given path in a digraph task.

**Definition 2** [11, 24, 25] For a task \( \tau \), the maximal cumulative execution request from a path \( \pi = (v_1, \ldots, v_l) \) in \( D(\tau) \) within any time interval of length \( t \) is defined as its request function \( \tau.r.f_\pi(t) \), i.e.,

\[
\tau.r.f_\pi(t) = \max \{ e(\pi') | \pi' \text{ is a prefix of } \pi \text{ and } p(\pi') < t \}
\]

where \( e(\pi) = \sum_{i=1}^{l} e(v_i) \) and \( p(\pi) = \sum_{i=1}^{l-1} p(v_i, v_{i+1}) \).

**Definition 3** [11] For a task \( \tau \), the maximal amount of executed load from a path \( \pi = (v_1, \ldots, v_l) \) in \( D(\tau) \) within any time interval of length \( t \) is defined as its interference function \( \tau.i.f_\pi(t) \), i.e.,

\[
\tau.i.f_\pi(t) = \max \{ ee(\pi') | \pi' \text{ is a prefix of } \pi \text{ and } p(\pi') < t \}
\]

where \( ee(\pi) = \sum_{i=1}^{l} e(v_i) + \min(e(v_l), \max(0, t - p(\pi))) \) and \( p(\pi) = \sum_{i=1}^{l-1} p(v_i, v_{i+1}) \).

Moreover, the demand function \( \text{df} \) is defined for tasks with l-MAD property [19]. It imposes a stronger requirement than \( \text{rf} \) and \( \text{if} \) functions, that the job deadlines are within the given time interval.

**Definition 4** [19] For a task \( \tau \) with l-MAD property, its demand function \( \tau.d.f_\pi(t) \) is defined as the maximal cumulative execution demand by its jobs from a path \( \pi = (v_1, \ldots, v_l) \) in \( D(\tau) \) that have their release times and deadlines within any time interval of length \( t \), i.e.,

\[
\tau.d.f_\pi(t) = \max \{ e(\pi') | \pi' \text{ is a prefix of } \pi \text{ and } d(\pi') \leq t \}
\]

where \( e(\pi) = \sum_{i=1}^{l} e(v_i) \) and \( d(\pi) = \sum_{i=1}^{l-1} p(v_i, v_{i+1}) + d(v_l) \).
Schedulability Analysis of Real-Time Digraph Tasks Scheduled with Static Priority

By the definitions, for any $t \leq 0$, $\tau.rf_\pi(t) = \tau.if_\pi(t) = \tau.df_\pi(t) = 0$. The rf function is a non-decreasing staircase function whose horizontal segments are left-open and right-closed. The if function is a continuous non-decreasing slanted staircase function, where the slope of each segment is either 0 or 1 [11]. Meanwhile, the df function is also a non-decreasing staircase function like rf, but its horizontal segments are left-closed and right-open. It is easy to verify that the if function can be derived from rf by replacing each vertical segment with a slanted segment. For simplicity, we call such a relationship as vertical-to-slope-one replacement in the rest of the paper.

**Theorem 2** For the rf and if functions of the same path $\pi = \{v_1, \ldots, v_l\}$ in the task $\tau$, there exists a vertical-to-slope-one replacement relationship defined as follows:

for any vertical segment of rf that $\tau.rf_\pi(t^+) = \tau.rf_\pi(t) + k$ where $k > 0$ (4) it must be $\tau.if_\pi(t) = \tau.rf_\pi(t)$, and $\forall t' \in [0, k], \tau.if_\pi(t + t') = \tau.if_\pi(t) + t'$ (5)

**Proof.** The proof is straightforward. Simply speaking, by the definition of rf, for any $t$ satisfying Equation (4), there must exist a prefix $\pi'$ of $\pi$ such that $t = p(\pi')$ and $k = e(v_j)$, where $v_j$ is the last vertex in $\pi'$. With the assumption in Equation (1), this implies that $\tau.if_\pi(t) = \tau.rf_\pi(t) = \sum_{i=1}^{j-1} e(v_i)$. In addition, by the definition of if, it is $\forall t' \in [0, k], \tau.if_\pi(t + t') = \tau.if_\pi(t) + t'$. □

However, such a relationship does not hold for their task-level abstractions (rbf and ibf). We provide a counterexample in Section 3.1.2.

**Example 2** As an example, Figure 2 shows the rf, if, and df functions for a path $\pi = (v_1, v_2)$ in the digraph of Figure 1. It is easy to verify that the rf and if functions satisfy Theorem 2.

3.1.2 Task-level abstraction

To derive faster analysis techniques, the above functions are abstracted and defined over a given task (instead of each path in the task). Below we define these new functions, namely request bound function (rbf), interference bound function (ibf), and...
demand bound function (dbf). In short, rbf (resp. ibf, dbf) can be regarded as the closest upper envelope of the corresponding path-level functions rf (resp. if, df) for all the paths.

**Definition 5** [5] For a task \( \tau \), the maximum cumulative execution requests by its jobs that have their release times within any time interval of length \( t \) is defined as its request bound function \( \tau . rbf(t) \), i.e.,

\[
\forall t \geq 0, \tau . rbf(t) = \max_{\pi \in P(\tau)} \{ \tau . rf_{\pi}(t) \}.
\]  

(6)

**Definition 6** [11] For a task \( \tau \), the maximum amount of executed load by its jobs that have their release times within any time interval of length \( t \) is defined as its interference bound function \( \tau . ibf(t) \), i.e.,

\[
\forall t \geq 0, \tau . ibf(t) = \max_{\pi \in P(\tau)} \{ \tau . if_{\pi}(t) \}.
\]  

(7)

Intuitively, rbf and ibf quantify the maximum cumulative execution request and those that are executed, respectively. dbf measures the amount of execution time from jobs that are released and must be completed within a given time interval. Similar to rf, rbf is a non-decreasing staircase function whose horizontal segments are left-open and right-closed, while ibf inherits the property of if as a continuous slanted staircase function. Like df, dbf is a staircase function where the horizontal segments are left-closed and right-open.

**Example 3** Figure 3 shows the rbf, ibf, and dbf functions for the digraph task in Figure 1. As in the example, in general rbf and ibf do not satisfy the vertical-to-slope-one replacement relationship. This is because for a given \( t \), the path contributing to rbf may be different from that to ibf. Consider time \( t \in (2, 3) \), \( \tau . rbf(t) = \tau . rf_{(v_3, v_1)}(t) = 3.0 \). However, \( \tau . if_{(v_3, v_1)}(t) = 1 + (t - 2) = t - 1 \), which is smaller than the if function of path \( (v_1, v_2) \): \( \tau . if_{(v_1, v_2)}(t) = 2 \).

By definition, the rbf, ibf and dbf functions can be calculated by searching the maximum among \( rf_s, if_s \) and \( df_s \) of all the possible paths, respectively. However, this is unscalable as the number of paths (hence the complexity) is exponential to the task graph size. Dynamic programming techniques can be used for an efficient calculation of these functions [20], and the algorithm complexity is polynomial to the time interval length \( t \).

To further speed up the calculation, especially for large \( t \), Zeng and Di Natale [28, 29] prove the linear periodicity of rbf (and dbf), shown in the following theorem.
Theorem 3 [28,29] The rbf and dbf functions of a generic digraph task $\tau$ are almost linear periodic, i.e., there exist a real number $q$ and a pair of integers $r$ and $p$ such that

$$\begin{align*}
\forall t > r, \quad &\tau.rbf(t + p) = \tau.rbf(t) + p \cdot q \\
\forall t > r, \quad &\tau.dbf(t + p) = \tau.dbf(t) + p \cdot q
\end{align*}$$

In the above equation, $r$, $p$, and $q$ are defined as the linear defect, linear period, and linear factor, respectively. It is shown that the linear factor is the same as the task’s utilization [28, 29], i.e.,

$$q = \lambda(\tau)$$

Theorem 3 allows the calculation of rbf and dbf functions with an asymptotical complexity of constant time with respect to the length of the time interval. Furthermore, the task graph transformation used to prove the linear periodicity also allows to derive tight linear bounds on rbf and dbf,

$$\begin{align*}
\forall t > 0, \quad &\tau.rbf(t) \leq C^{rbf} + \lambda(\tau) \cdot t \\
\forall t > 0, \quad &\tau.dbf(t) \leq C^{dbf} + \lambda(\tau) \cdot t
\end{align*}$$

where $C^{rbf}$ and $C^{dbf}$ are constants.

In Section 4, we prove that for any digraph real-time task, its ibf has a similar property of linear periodicity as the rbf and dbf functions, and provide a similar linear upper bound on ibf.
3.2 Schedulability Analysis under Frame Separation Property

Once these functions have been computed, we can analyze the schedulability of the digraph systems. We start with the case that the tasks satisfy the frame separation property. Under this property, for any task the absolute deadline of any vertex is no larger than the release time of the next vertex. Hence, the schedulability of a vertex is not affected by other vertices from the same task, assuming that the other vertices are schedulable. For a vertex \( v \) in the task \( \tau_v \) under analysis, let \( f_1; \ldots; f_n \) be the set of higher priority tasks, \( \Pi(\Gamma) \) be the set of all path combinations from tasks in \( \Gamma \), and \( \sigma = (\pi_1, \ldots, \pi_n) \) denote an element in \( \Pi(\Gamma) \). The exact response time of \( v \) is the maximum among all the path combinations in \( \Pi(\Gamma) \), as in the following equation [11]

\[
R(v, \Gamma) = \max_{\sigma \in \Pi(\Gamma)} \left\{ \min_{t > 0} \left\{ t|e(v) + \sum_{\pi_i \in \sigma} \tau_i \cdot r_f(t) \leq t \right\} \right\}
\]

\[
= \max_{\sigma \in \Pi(\Gamma)} \left\{ \min_{t > 0} \left\{ t|e(v) + \sum_{\pi_i \in \sigma} \tau_i \cdot i_f(t) \leq t \right\} \right\} \tag{11}
\]

In Equation (11), there are two alternatives, one is to use \( r_f \), the other uses \( i_f \), as these two functions are equivalent for the computation of response times [11]. The exact schedulability analysis of a task can be checked by testing whether all its vertices satisfy the condition \( R(v, \Gamma) \leq d(v) \).

As in Equation (11), the exact analysis requires an exhaustive check of all possible combinations of request functions (and consequently, all path combinations) from the higher priority tasks. It is known that the number of paths in a digraph is exponential to the number of nodes. To improve the analysis efficiency, an iterative refinement method is proposed [19], where abstraction of the paths is first used to check the schedulability and the refinement is only performed if the schedulability condition is not satisfied. However, the worst case complexity remains to be the same.

Alternatively, approximate schedulability analysis is proposed to find sufficient only conditions. Similar to Equation (11), for a vertex \( v \) and its interfering task set \( \Gamma \), its approximate response times can be derived as the following equations [11]:

\[
R_{RBF}(v, \Gamma) = \min_{t > 0} \left\{ t|e(v) + \sum_{\pi_i \in \Gamma} \tau_i \cdot r_f(t) \leq t \right\}
\]

\[
R_{IBF}(v, \Gamma) = \min_{t > 0} \left\{ t|e(v) + \sum_{\pi_i \in \Gamma} \tau_i \cdot i_f(t) \leq t \right\} \tag{12}
\]

Equation (12) gives the response times \( R_{RBF}(v, \Gamma) \) and \( R_{IBF}(v, \Gamma) \) of the vertex \( v \), calculated using \( r_f \) and \( i_f \) respectively. It avoids the path combination problem in the exact response time analysis, as each higher priority task \( \tau_i \) is characterized with a single \( r_f \) or \( i_f \) function (as opposed to one for each path in the exact analysis) [11]. A vertex \( v \) is deemed schedulable if \( R_{RBF}(v, \Gamma) \leq d(v) \) or \( R_{IBF}(v, \Gamma) \leq d(v) \).
In addition, since \( \forall t \geq 0, \tau \cdot ibf(t) \leq \tau \cdot rbf(t) \), the following inequality always holds [11]

\[
R(v, \Gamma) \leq R_{IBF}(v, \Gamma) \leq R_{RBF}(v, \Gamma)
\]

The following two concepts of the approximate ratio [7] and speedup factor [11, 12] can measure the approximation quality of any approximate analysis compared to the exact response time.

**Definition 8** [7] A response time analysis method \( A \) has an approximation ratio of \( c \), if the response time \( R_A \) estimated by \( A \) can be bounded by \( c \)-times of the exact response time \( R \), i.e., \( R \leq R_A \leq c \cdot R \). If such a constant \( c \) does not exist, the method \( A \) is said to have no approximate ratio.

**Definition 9** [11, 12] A response time analysis method \( A \) has a speedup factor of \( u \), if the response time estimated by \( A \) on a speed-\( u \) processor, denoted as \( R^u_A \), is bounded by its exact response time \( R \), i.e., \( R^u_A \leq R \). If such a constant \( u \) does not exist, the method \( A \) is said to have no speedup factor.

[11] demonstrates that the \( rbf \) and \( ibf \) analyses have bounded speedup factors as follows.

**Theorem 4** [11] The \( rbf \) analysis has a speedup factor of 2, while the \( ibf \) analysis has a speedup factor of \( 1 + \sqrt{m^2 - m/m} \), where \( m \) is the number of higher priority tasks.

We now prove that these two analysis techniques have no approximate ratio.

**Theorem 5** The \( rbf \) and \( ibf \) analyses have no approximate ratio.

**Proof.** For any given approximation ratio \( c > 1 \), we construct a counterexample task system as follows. It consists of a task under analysis with only one vertex \( v \) and the higher priority task set containing two tasks \( \Gamma = \{\tau, \tau'\} \). The WCET of \( v \) is \( e(v) = 1/k \) where \( k = 2c \). Figures 4(a)–4(b) illustrate the digraphs of the two interfering tasks \( \tau \) and \( \tau' \) respectively.

Figure 4(c) shows the total request bound function (i.e., \( \tau \cdot rbf(t) + \tau' \cdot rbf(t) + e(v) \)) and interference bound function (i.e., \( \tau \cdot ibf(t) + \tau' \cdot ibf(t) + e(v) \)). By applying Equation (12), \( R_{RBF}(v, \Gamma) (R_{IBF}(v, \Gamma)) \) is the first intersection for the two curves representing the total request (interference) bound function and the line with slope 1. From Figure 4(c), we have

\[
R_{RBF}(v, \Gamma) = R_{IBF}(v, \Gamma) = k + 2
\]

It is easy to verify that the exact response time is \( R(v, \Gamma) = 2 + 1/k \), where the contributing path combination from \( \Gamma \) is \( ((v_1), (v'_1)) \). Hence, the approximation ratio is

\[
\frac{R_{RBF}(v, \Gamma)}{R(v, \Gamma)} = \frac{R_{IBF}(v, \Gamma)}{R(v, \Gamma)} = \frac{k + 2}{2 + \frac{1}{k}} > \frac{k}{2} = c
\]

\( \square \)
Fig. 4 An example demonstrating that the rbf and ibf analyses have no approximate ratio. The vertex $v$ under analysis has WCET $e(v) = 1/k$. The higher priority task set $I = \{\tau, \tau'\}$ are illustrated in Figures 4(a)–4(b). Here we omit the deadlines in $\tau$ and $\tau'$ as they are irrelevant for the schedulability of $v$.

3.3 Schedulability Analysis under l-MAD Property

Under l-MAD Property, the response time of a vertex $v$ can be affected by its preceding vertex $u$ \((u, v) \in E\) even if $u$ is assumed schedulable, as $u$ can continue its execution when $v$ is released. Therefore, Equation (11) and (12) do not hold. In the following, we summarize the approximate schedulability analysis for such tasks. The exact schedulability analysis is provided in Section 3.4 for tasks with arbitrary deadlines, which is a more general case than tasks with l-MAD property.

The following theorems present the sufficient only conditions for checking the schedulability of task sets under the l-MAD property by using the task level workload functions, i.e., rbf, ibf, and dbf.

**Theorem 6** \([5, 29]\) For a task $\tau$ and its interfering task set $\Gamma$ under the l-MAD property, $\tau$ is schedulable if

$$\forall t \geq 0, \exists t', \text{ s.t. } \tau.dbf(t) + \sum_{\tau_i \in \Gamma} \tau_i.rbf(t') \leq t'$$

**Theorem 7** For a task $\tau$ and its interfering task set $\Gamma$ under the l-MAD property, $\tau$ is schedulable if

$$\forall t \geq 0, \exists t', \text{ s.t. } \tau.dbf(t) + \sum_{\tau_i \in \Gamma} \tau_i.ibf(t') \leq t'$$

Theorem 7 is a trivial extension that can be proved in the same way as Theorem 6. We omit the proof here.

Generally speaking, it is impractical to check all $t \geq 0$ for Theorem 6 and 7. However, an upper bound $t_f$ can be derived on the set of time instants $t$ to check the schedulability condition. For a task $\tau$ with the interference task set $\Gamma$, if the total
utilization is less than 1 (otherwise, the system cannot be scheduled), [29] gives an
upper bound $t_f$ as follows
\[ t_f = \frac{\tau v C^{dbf} + \sum_{\tau_i \in F} \tau_i C^{rbf}}{1 - \lambda(\tau) - \sum_{\tau_i \in F} \lambda(\tau_i)} \]  
(15)
It leverages the linear upper bounds on $rbf$ and $dbf$ in Equation (10). In Section 5, we
discuss the linear upper bound on the interference bound function, which can further
improve $t_f$ compared to Equation (15).

3.4 Schedulability Analysis for Tasks with Arbitrary Deadlines

For tasks with arbitrary deadlines, the analysis in Section 3.3 is not applicable as
it is not safe. Without the l-MAD property, a vertex $v$ may have a smaller absolute
deadline than those of the previously released jobs in the same task. However, the
execution of $v$ shall wait for the finish of the previous jobs regardless of their deadline
settings. On the other hand, the $df$ and $dbf$ functions cannot accurately capture such
possible interferences from jobs of the same task. We provide an example below.

Example 4 Figure 5 shows an example that the two theorems, Theorem 6 and Theo-
rem 7, do not hold for the case of arbitrary deadline. $\forall t \in [2, 10)$, $\tau v, df(t) = 1$ as
the deadline of $v_2$ is larger than $t$. Hence, the interference of $v_2$ to $v_3$ is not well cap-
tured in the $dbf$ function. Consequently, we can verify that $\forall t \in [2, 10)$, the schedu-
ability condition in (13) or (14) is always satisfied (with $t' = 2$). In general, $\tau_2$ is
deemed schedulable for any $t \geq 0$ by Theorem 6 and Theorem 7. This gives a false
positive on the schedulability of task $\tau_2$. By the exact analysis in Equation (18) devel-
oped below, the exact response time of $v_3$ is 3.5, which exceeds the deadline. Hence,
task $\tau_2$ is unschedulable.

In the following, we discuss how to derive safe schedulability analysis for tasks
with arbitrary deadlines. Consider a vertex $v$ in task $\tau_v$, let $\pi_v$ denote an arbitrary
path in $D(\tau_v)$ that ends at $v$. We use the same notations $F$ (higher priority task set),
$II(F)$ (path combinations of $F$), and $\sigma$ (an element in $II(F)$) as in Section 3.2. Thus,
we can express the exact response time of $\pi_v$ with path combination $\sigma$ as follows.

\[ R(\pi_v, \sigma) = \min_{t > p(\pi_v)} \left\{ t - p(\pi_v) + \sum_{\pi_i \in \sigma} \tau_i \cdot rf_{\pi_i}(t) \leq t \right\} \]
\[ = \min_{t > p(\pi_v)} \left\{ t - p(\pi_v) + \sum_{\pi_i \in \sigma} \tau_i \cdot if_{\pi_i}(t) \leq t \right\} \]  
(16)
Similar to the case with frame separation, it is easy to verify the equivalence of the two functions \( r_f \) and \( i_f \) in deriving the equation.

The idea of Equation (16) is to find the first continuously busy interval \([0, t]\) (namely, the busy period introduced in [13]). There are two cases:

(a) If \( t > p(\pi_v) \), it means that the busy period ends after the release time \( p(\pi_v) \) of the vertex \( v \). \( v \) completes the execution at \( t \), thus the response time of \( v \) can be calculated as \( t - p(\pi_v) \).

(b) Otherwise, \( v \) must arrive after the busy period ends. The released jobs of \( \sigma \) and \( \pi_v \) in the busy period cannot interfere the execution of \( v \). Note that \( R(\pi_v, \sigma) \) is negative in this case.

Case (b) above, although included in the computation as it cannot be easily ruled out a priori, will not determine the actual response time of \( v \) (as it should be). This is because of the following two max operators over all the path combinations in \( \Pi(\Gamma) \) (Equation (17)), and over all possible \( \pi_v \) (Equation (18)). The response time \( R(\pi_v, \Gamma) \) for \( \pi_v = (v) \) is guaranteed to be positive, hence dominating those negative response times given in case (b).

The response time of \( \pi_v \) is the maximum among all the path combinations in \( \Pi(\Gamma) \), as in the following equation.

\[
R(\pi_v, \Gamma) = \max_{\sigma \in \Pi(\Gamma)} R(\pi_v, \sigma) \tag{17}
\]

The exact response time of \( v \) can be expressed by enumerating all the paths ending at itself, described as below.

\[
R(v, \Gamma) = \max_{\pi_v \in \mathcal{D}(\tau_v)} R(v, \Gamma) \tag{18}
\]

Intuitively, Equation (18) examines all the possible busy periods within which \( v \) is released. Similar to [13], the largest response time of any vertex \( v \) occurs when \( \tau_v \) and its interfering task set \( \Gamma \) arrive at the same time. Hence, this equation can correctly calculate the exact response time of \( v \). Moreover, it is a generalization of Equation (11) in Section 3.2: under frame separation property \( R(v, \Gamma) \) takes the maximal value when \( \pi_v = (v) \). Obviously Equation (18) has an even higher complexity than (11).

**Example 5** We now apply the exact analysis to the example in Figure 5. By Equation (18), the exact response time of \( v_3 \) is 3.5 where the contributing worst case path from \( \tau_2 \) is \( \pi_v = (v_2, v_3) \) and the worst case path from \( \tau_1 \) is \( (v_1, v_1, v_1) \). Hence, the deadline of \( v_3 \) is violated, and task \( \tau_2 \) is unschedulable.

To reduce the complexity of the schedulability analysis, we develop approximate response times using the \( rbf \) and \( ibf \) functions to abstract the interferences from higher priority tasks \( \Gamma \), expressed as follows.

\[
R_{RBF}(v, \Gamma) = \max_{\pi_v \in \mathcal{D}(\tau_v)} R_{RBF}(v, \Gamma) = \max_{\pi_v \in \mathcal{D}(\tau_v)} \left\{ \min_{t > 0} \left\{ t - p(\pi_v) R_f(\pi_v)(t) + \sum_{\pi_i \in \Gamma} \tau_{i,RBF}(t) \leq t \right\} \right\} \tag{19}
\]
\[ R_{IBF}(v, \Gamma) = \max_{\pi_v \in \mathcal{D}(\tau_v)} \left\{ \min_{t > 0} \left\{ t - p(\pi_v) \right\} \right\} \]

In addition, it is safe to restrict the involved path \( \pi_v \) with \( p(\pi_v) \leq L_{RBF} \), where \( L_{RBF} \) is the smallest \( t > 0 \) satisfying \( \sum_{\tau_i \in \Gamma \setminus \tau_v} \tau_i ibf(t) \leq t \). Essentially \( L_{RBF} \) is derived by using \( \tau_v ibf \) as a safe upper bound for the rf of any \( \pi_v \). Since for any path \( \pi_v \) with \( p(\pi_v) > L_{RBF} \), the preceding vertices of \( v \) is guaranteed to finish before the release time \( p(\pi_v) \) of \( v \), hence they have no impact on the execution of \( v \). The same reason applies to derive the bound \( L_{IBF} \), the smallest \( t > 0 \) satisfying \( \sum_{\tau_i \in \Gamma \setminus \tau_v} \tau_i ibf(t) \leq t \).

4 Demonstrating the periodicity of the ibf

In this section, we demonstrate the linear periodicity of ibf. The proof relies on a task graph transformation to a grained digraph real-time task (GDRT). The task graph transformation is different from that for rbf. Simply speaking, in the proof of the linear periodicity for rbf [29], the transformed graph unit digraph real-time task (UDRT) preserves the same rbf as the original digraph. However, as we exemplify earlier in Example 3, in general there is no direct relation between a digraph’s rbf and ibf functions. Hence, we need a new way to prove the linear periodicity of ibf.

We first describe the task graph transformation to a grained digraph real-time task (GDRT) from the original digraph. The transformed GDRT has the same ibf as the original digraph. GDRT is a special kind of digraph task, hence its rbf is also linear periodic. Moreover, we can prove for any GDRT its ibf equals its rbf when the time length \( t \) is an integer. This is sufficient to guarantee the vertical-to-slope-one replacement relationship between ibf and rbf, which allows us to infer the linear periodicity of the GDRT’s ibf and consequently that of the original digraph.

4.1 Task Transformation

For a generic digraph task, we transform it to another one, called a grained digraph real-time task (GDRT). In GDRT, all the inter-release times are equal to 1 (the same as UDRT), and the WCET of each node is either 0 or 1. More specifically, for a digraph task \( \mathcal{D}(\tau) = (\mathcal{V}, \mathcal{E}) \), we generate a corresponding GDRT \( \mathcal{D}'(\tau') = (\mathcal{V}', \mathcal{E}') \) by the following rules:

- Each vertex \( v_i \in \mathcal{V} \) corresponds to \( k_i \) vertices \( v_{i,1}, v_{i,2}, ..., v_{i,k_i} \) in \( \mathcal{V}' \), where \( k_i = \max \{ 1, \max_{v_j : (v_i, v_j) \in \mathcal{E}} k_{i,j} \} \), and \( k_{i,j} = p(v_i, v_j) \).
- Nodes \( v_{i,1}, ..., v_{i,m_i} \) in the GDRT are labeled with an execution time of one, where \( m_i = e(v_i) \). All the other vertices \( v_{i,m_i+1}, ..., v_{i,k_i} \) have an execution time of zero.
For every node \(v_{i,k}(1 \leq k \leq k_i-1)\), an edge \((v_{i,k}, v_{i,k+1}) \in E'\) is added to connect these nodes in a chain topology.

- Each edge \((v_i, v_j) \in E\) corresponds to the edge \((v_{i,k_i}, v_{j,1}) \in E'\).
- All edges in \(E'\) are labeled with a minimum separation time \(\text{one}\).
- Deadlines can be assigned arbitrarily, as we are only interested in the \(rbf\) and \(ibf\) functions of the transformed graph.

For a digraph task \(\tau\), its transformed UDRT in [28,29] and the GDRT in this paper are isomorphic, in the sense that they have the same graph structure. Their edge labels (the job inter-release times) are also exactly the same (all equal to one). The only difference lies in the distribution of the worst case execution times: for a vertex \(v_i\) in \(D(\tau)\), its execution time \(e(v_i)\) is distributed to the vertices \(v_{i,1}, \ldots, v_{i,e(v_i)}\) in the GDRT, while in the UDRT it is aggregated at \(v_{i,1}\).

**Example 6** For the digraph task \(\tau\) in Figure 1, the transformed GDRT is shown in Figure 6. \(v_1\) in \(\tau\) is transformed to four vertices \(v_{1,1}, v_{1,2}, v_{1,3},\) and \(v_{1,4}\) in \(\tau'\), \(v_2\) is...
transformed to $\nu_2, \nu_2, \nu_2, \nu_2, \nu_3, \nu_3$, and $\nu_3$ to $\nu_3$ and $\nu_3$. The vertices $\nu_1, \nu_1, \nu_2, \nu_2, \nu_2, \nu_2$, and $\nu_3, \nu_3, \nu_3$ are each assigned with a WCET of 1, while all the other vertices have an execution time of 0. The minimum inter-release times are all equal to 1 and, along with the node deadlines, are omitted in the figure. As a comparison, Figure 7 illustrates the transformed UDRT of the digraph task in Figure 1. It is easy to verify that the two graphs are isomorphic.

The following property of the transformation allows us to use the transformed GDRT to study the ibf of a digraph task.

Lemma 8 The interference bound functions of a digraph task $D(\tau)$ and its transformed GDRT $D'(\tau')$ are the same.

Proof. (a) $\forall t \geq 0$, $\tau.\text{ibf}(t) \leq \tau'.\text{ibf}(t)$.

Let $\pi = (v_1, \ldots, v_l)$ be an arbitrary path in $D(\tau)$. We construct a corresponding path $\pi'$ by replacing each vertex $v_i$ except the last one $v_l$ in $\pi$ with a sequence of vertices $\nu_{i,k}$ for $k = 1, \ldots, k_{i+1}$, $p(v_l, v_{i+1})$. The last vertex $v_l$ in $\pi$ is replaced with a sequence of vertices $\nu_{l,k}$ for $k = 1, \ldots, m_l = e(v_l)$. Because of the way $\pi'$ is constructed, $\pi'$ is a legal path in $D'(\tau')$. In the following, we prove that for any $t \geq 0$, $\tau.\text{if}_{\pi}(t) \leq \tau'.\text{if}_{\pi'}(t)$, hence $\tau.\text{ibf}(t) \leq \tau'.\text{ibf}(t)$.

For any $t \geq 0$, there exists a prefix of $\pi$, denoted as $\hat{\pi} = (v_1, \ldots, v_j)$, such that

$$p(\hat{\pi}) = \sum_{i=1}^{j-1} p(v_i, v_{i+1}) = \sum_{i=1}^{j-1} k_{i,i+1} < t$$

We find a corresponding prefix of $\pi'$, denoted as $\hat{\pi}' = (\nu_{1,1}, \ldots, \nu_{1,k_1}, \ldots, \nu_{j,1}, \ldots, \nu_{j,\min(m_j,t-p(\pi))})$

It can be verified that

$$p(\hat{\pi}') \leq \sum_{i=1}^{j-1} k_{i,i+1} + (t - p(\hat{\pi}) - 1) = t - 1 < t$$

Furthermore, the if functions of $\hat{\pi}$ and $\hat{\pi}'$ satisfy

$$\tau'.\text{if}_{\hat{\pi}'}(t) = \sum_{i=1}^{j-1} \sum_{k=1}^{k_{i+1}} e(\nu_{i,k}) + \min_{k=1}^{\min(m_j,t-p(\hat{\pi}))} e(\nu_{j,k})$$

$$= \sum_{i=1}^{j-1} e(v_i) + \min_{j} e(v_j)$$

$$= \sum_{i=1}^{j-1} e(v_i) + \max(0, t - p(\hat{\pi}))$$

$$= \tau.\text{if}_{\hat{\pi}}(t)$$

This implies $\tau.\text{if}_{\pi}(t) \leq \tau'.\text{if}_{\pi'}(t)$, hence $\tau.\text{ibf}(t) \leq \tau'.\text{ibf}(t)$.

(b) $\forall t \geq 0$, $\tau'.\text{ibf}(t) \leq \tau.\text{ibf}(t)$.

Assume any path $\hat{\pi}'$ in $D'(\tau')$

$$\hat{\pi}' = (\nu_{1,1}, \ldots, \nu_{1,k_{1,2}}, \nu_{2,1}, \ldots, \nu_{2,k_{2,3}}, \ldots, \nu_{j,1}, \ldots, \nu_{j,y})$$
where $1 \leq x \leq k_{1,2}$ and $1 \leq y \leq k_j$. We extend the last vertex sequence $(\nu_{j,1}, \ldots, \nu_{j,y})$ to at least of length $e(v_j)$, resulting the new path below

$$\pi' = (\nu_{1,x}, \ldots, \nu_{1,k_{1,2}}, \nu_{2,1}, \ldots, \nu_{2,k_{2,2}}, \ldots, \nu_{j,1}, \ldots, \nu_{j,\max\{y,e(v_j)\}})$$

Obviously $\tau'.if_\pi(t) \leq \tau'.if_{\pi'}(t)$, $\forall t \geq 0$.

For the first vertex sequence $\pi_1' = (\nu_{1,x}, \ldots, \nu_{1,k_{1,2}})$, we discuss two cases.

**Case 1**: If $m_1 = e(v_1) < x \leq k_{1,2}$, it means any vertex in $\pi_1'$ has a zero WCET. We construct a path $\pi$ in $D(\tau)$

$$\pi = (v_2, \ldots, v_j)$$

where the vertices in $\pi_1'$ are omitted, and the other vertex sequence $(\nu_{i,1}, \ldots, \nu_{i,k_{i,i+1}})$ ($2 \leq i \leq j$) is replaced with the vertex $v_i$.

As illustrated in Figure 8(a), $\tau'.if_\pi(t)$ can be computed by simply shifting $\tau.if_\pi(t)$ to the right by $k_{1,2} - x + 1$. Hence, it is

$$\forall t \geq 0, \quad \tau'.if_\pi(t) \leq \tau'.if_{\pi'}(t) \leq \tau.if_\pi(t)$$

**Case 2**: Otherwise, in $\pi$ we use vertex $v_1$ to replace $\pi_1'$:

$$\pi = (v_1, v_2, \ldots, v_j)$$

In this case, $\tau'.if_{\pi'}(t)$ can be computed by simply shifting $\tau.if_\pi(t)$ both down and to the left by $x - 1$, as illustrated in Figure 8(b). Hence, it also holds that $\forall t \geq 0, \tau'.if_{\pi'}(t) \leq \tau.if_\pi(t)$.

Combining the two cases, we have $\tau'.if_{\pi'}(t) \leq \tau'.if_{\pi'}(t) \leq \tau.if_\pi(t)$. Since this is satisfied for any path $\hat{\pi}'$, it must be that $\tau'.ibf(t) \leq \tau.ibf(t)$. $\Box$

### 4.2 Periodicity of ibf

In this subsection, we prove the periodicity of the ibf by demonstrating the equality between the ibf and rbf functions at any integer time instant ($\forall t \in \mathbb{N}$) for any GDRT. This allows us to demonstrate the linear periodicity of ibf of the transformed digraph, and consequently of the original digraph.
Lemma 9 For any GDRT $\mathcal{D}'(\tau')$, we have the following property:

$$\forall t \in \mathbb{N}, \tau'.ibf(t) = \tau'.rbf(t).\quad (21)$$

Proof. By induction on $t$.

Initial Step. This is trivial: for $t = 0$, by the definitions of $rbf$ and $ibf$, $\tau'.ibf(0) = \tau'.rbf(0) = 0$.

Inductive Step. For $t = k$, we assume that $\tau'.ibf(k) = \tau'.rbf(k)$. Since for any vertex $\nu$ in the GDRT, $e(\nu) \in \{0, 1\}$, $\tau'.rbf(k + 1)$ can only take two possible values: $\tau'.rbf(k + 1) = \tau'.rbf(k)$ or $\tau'.rbf(k + 1) = \tau'.rbf(k) + 1$.

Case 1: $\tau'.rbf(k + 1) = \tau'.rbf(k)$. By the definitions of $rbf$ and $ibf$, it holds that $\tau'.ibf(k + 1) \leq \tau'.rbf(k + 1)$. Also, we notice that the $ibf$ is a non-decreasing function, hence $\tau'.ibf(k) \leq \tau'.ibf(k + 1)$. Combining the above facts, we have

$$\tau'.rbf(k) = \tau'.ibf(k) \leq \tau'.ibf(k + 1) \leq \tau'.rbf(k + 1) = \tau'.rbf(k)$$

Thus, the inductive step is proved for both of the above two cases. □

We now can see that the $ibf$ and $rbf$ functions of any GDRT satisfy the relationship of vertical-to-slope-one replacement. More specifically, consider a vertical segment in $rbf$ at time $k \in \mathbb{N}$: $\tau'.rbf(k + 1) = \tau'.rbf(k) + 1$. Due to the following two reasons, the corresponding $ibf$ is a segment of slope 1, i.e., $\forall t \in [k, k + 1], \tau'.ibf(t) = \tau'.ibf(k) + (t - k)$.

(a) $\tau'.ibf(k + 1) = \tau'.ibf(k) + 1$ by Lemma 9;
(b) the slope of $ibf$ cannot be larger than 1 ($ibf$ cannot increase faster than the available CPU time), hence it has to be 1 in $[k, k + 1]$ to satisfy the relationship in (a).

With the assumption of the inductive step and the condition in Case 2, we have

$$\tau'.ibf(k + 1) = \tau'.ibf(k) + 1 = \tau'.rbf(k) + 1 = \tau'.rbf(k + 1)$$

Thus, the inductive step is proved for both of the above two cases. □
Example 7 Figure 9 illustrates the rbf and ibf of the original task $\tau$ and the rbf of the transformed GDRT $\tau'$. It can be seen that the rbf of the GDRT has a staircase shape with a vertical increase of 1. We can verify that the ibf and rbf functions of $\tau'$ satisfy the relationship of vertical-to-slope-one replacement.

It is worth noting that the original DRT (in Figure 1) do not satisfy Lemma 9, for example, at times $t = 1, 4, 7$. This is because unlike GDRT where each vertex only has an WCET equal to either 0 or 1, vertices of the original DRT in general have different worst case execution time values (e.g., $e(v_1) = 2$ and $e(v_2) = 2$ in Figure 1).

We now prove that the ibf function of a generic digraph task is almost linear periodic, leveraging the facts that the rbf function of its transformed GDRT is almost linear periodic (demonstrated by Theorem 3), and the equivalence of ibfs of the original digraph task and its transformed GDRT task (Lemma 8).

**Theorem 10** The ibf function of a generic digraph task $\tau$ is almost linear periodic, i.e., there exist a real number $q$ and a pair of integers $r$ and $p$ such that

$$\forall t > r, \quad \tau . ibf(t + p) = \tau . ibf(t) + p \cdot q.$$ \hspace{1cm} (22)

**Proof.** For the digraph task $\tau$, we construct its transformed GDRT task $\tau'$. By Theorem 3, there exist a real number $q'$ and a pair of integers $r'$ and $p'$ such that

$$\forall t > r', \quad \tau' . rbf(t + p') = \tau' . rbf(t) + p' \cdot q'$$ \hspace{1cm} (23)

We prove the periodicity of $ibf$ as in Equation (22), where $r = r' + 1$, $p = p'$, and $q = q'$. 
For any integer $t$ that is larger than $r'$, i.e., $t > r'$, $t \in \mathbb{N}$, by Lemma 9, we have the following relation:

$$\forall t > r' \wedge t \in \mathbb{N}, \quad \tau'.ibf(t + p) = \tau'.ibf(t) + p \cdot q \quad (24)$$

For the general case that $t$ is any real number, we establish a relationship between $\tau'.ibf(t)$ with those at the surrounding integer times $k$ and $k + 1$, where $k = \lfloor t \rfloor$, and $\alpha = t - k$.

$$\tau'.ibf(t) = (1 - \alpha) \cdot \tau'.ibf(k) + \alpha \cdot \tau'.ibf(k + 1) \quad (25)$$

We prove it by considering two cases. As in the proof of Lemma 9, $\tau'.ibf(k + 1)$ can only have two possible values: 0 or 1.

Case 1: $\tau'.ibf(k) = \tau'.ibf(k + 1)$. Then it must be $\tau'.ibf(t) = \tau'.ibf(k) = \tau'.ibf(k + 1)$ due to the non-decreasing property of $ibf$, and (25) is satisfied.

Case 2: $\tau'.ibf(k + 1) = \tau'.ibf(k) + 1$. By the definition of $ibf$, its slope cannot be larger than 1, and we have $\tau'.ibf(k + \alpha) \leq \tau'.ibf(k) + \alpha$. Hence, it must be $\tau'.ibf(k + \alpha) = \tau'.ibf(k) + \alpha$ to be able to increase by 1 from time $k$ to time $k + 1$. In this case, (25) is also satisfied.

With (25) established, we can derive that (22) is true. $\forall t > r' + 1$, since $k = \lfloor t \rfloor > r'$, Equation (23) applies. Hence,

$$\tau'.ibf(t + p')$$

$$= \alpha \cdot \tau'.ibf(k + 1 + p') + (1 - \alpha) \cdot \tau'.ibf(k + p')$$

(by Equation (25))

$$= \alpha \cdot \tau'.rbf(k + 1 + p') + (1 - \alpha) \cdot \tau'.rbf(k + p')$$

(by Lemma 8)

$$= \alpha \cdot (\tau'.rbf(k + 1) + p' \cdot q') + (1 - \alpha) \cdot (\tau'.rbf(k) + p' \cdot q')$$

(by Equation (23))

$$= \alpha \cdot \tau'.rbf(k + 1) + (1 - \alpha) \cdot \tau'.rbf(k) + p' \cdot q'$$

(by Lemma 8)

$$= \tau'.ibf(t) + p' \cdot q'$$

(by Equation (25))

(26)

Example 8 As an example, the $ibf$ of the digraph in Figure 1 has the following linear periodicity (where $p = 3$, $q = \frac{2}{3}$, and $r = 16$):

$$\forall t > 16, \quad \tau.ibf(t + 3) = \tau.ibf(t) + 2$$

4.3 Computing the periodicity parameters for strongly connected task graphs

To use the periodicity property for speeding up the computation of $ibf$, we have to determine the periodicity parameters. We focus on the class of strongly connected task digraphs. This fits well with most applications of practical interests especially for safety-critical systems, because of the need to bring back the system to a safe state [29].

By the proof of Theorem 10, we can summarize the following useful property. It provides a direct relationship between the linear periodicity parameters for $ibf$ and $rbf$ in the GDRT.
Corollary 11 For any digraph task $\tau$ and its transformed GDRT $\tau'$, if the linear period, linear factor, and linear defect of $\tau'$ are $p$, $q$, and $r$, respectively, i.e.,

$$\forall t > r, \tau'.rbf(t + p) = \tau'.rbf(t) + p \cdot q$$

it holds that

$$\forall t > r + 1, \tau.ibf(t + p) = \tau.ibf(t) + p \cdot q$$

By Corollary 11, each periodicity parameter of $ibf$ for a digraph task can be estimated by the corresponding one of $rbf$ for its transformed GDRT task. Note that any GDRT is also a unit digraph task (UDRT) (introduced in [28, 29]), since all the edges in the GDRT are of unit-weight.

In [28, 29], we developed algorithms to efficiently calculate the linear periodicity parameters for the $rbf$ of a digraph task $D(\tau) = (V, E)$. We summarize the algorithms and their time complexity below.

- **Linear factor $q$**. The linear factor is essentially the maximum cycle mean of $D'(\tau)$, which can be computed using the algorithm in [26]. The time complexity is $O(|V||E| \log(|V|))$.

- **Linear period $p$**. The algorithm to compute the linear period operates through a few steps as detailed in [10, 29]. The time complexity is $O(|V|^3)$.

- **Linear defect $r$**. The linear defect can be computed over the original digraph $D(\tau)$ using the dynamic programming procedure proposed in [20]. The time complexity is $O(|V| + |E|(r + p))$.

These procedures are sensitive to the size of the digraphs. Hence, to calculate the linear periodicity parameters of $ibf$ for task $\tau$, it is inefficient through its transformed GDRT $\tau'$ (Corollary 11). $D'(\tau')$ is much larger than the original DRT $D(\tau)$, as each vertex $v_i$ in $D(\tau)$ corresponds to $k_i$ vertices in $D'(\tau')$. The algorithms for calculating the $rbf$ of $D'(\tau')$ will depend on the size of $D'(\tau')$, which is undesired.

In the following, we build up the relationship of the linear periodicity parameters between the $rbf$ and $ibf$ for a digraph task to avoid the complexity of operating on the transformed GDRT. Specifically, we leverage the known results among the digraph task $D(\tau)$ and its transformed GDRT $D'(\tau')$ and UDRT $D''(\tau'')$: (a) the $ibfs$ of $\tau$ and $\tau'$ are the same (Lemma 8); (b) the $rbfs$ of $\tau$ and $\tau''$ are equal [29]; and (c) the periodicity parameters of $rbf$ and $ibf$ of $\tau'$ satisfy Corollary 11. We make up the missing link by finding that the linear periodicity parameters of the $rbfs$ are the same for $\tau'$ and $\tau''$, as presented in Lemma 14. This allows to provide a simple relationship between the periodicity parameters of $ibf$ and $rbf$ for $\tau$ (Theorem 15). Hence, the efficient algorithms as listed above can be used, whose complexity does not depend on the size of the GDRT but on that of the original digraph.

We first consider a lemma for the $rbf$s of a pair of isomorphic paths from $\tau'$ and $\tau''$ respectively.

Lemma 12 For a generic digraph task $\tau$, consider a path $\pi'$ in the transformed GDRT task $\tau'$ and its isomorphic path $\pi''$ in the transformed UDRT task $\tau''$:

$$\pi' = (v_{t,1}, \ldots, v_{t,k_1,1}, v_{t+1,1}, \ldots, v_{t,2,1}, \ldots, v_{t,k_2,1}, \ldots, v_{t+1,1}, \ldots, v_{t-1,1,1}, \ldots, v_{t,1,1}, \ldots, v_{t-1,k_1,1}, v_{t,1,1}, \ldots, v_{t,1,y})$$

$$\pi'' = (v_{t,1}, \ldots, v_{t,k_1,1}, v_{t+1,1}, \ldots, v_{t,2,1}, \ldots, v_{t,k_2,1}, \ldots, v_{t+1,1}, \ldots, v_{t-1,1,1}, \ldots, v_{t,1,1}, \ldots, v_{t-1,k_1,1}, v_{t,1,1}, \ldots, v_{t,1,y})$$
where path \( \pi'' \) is derived from \( \pi' \) by simply replacing node \( \nu'_{i,j} \) with the corresponding node \( \nu''_{i,j} \). The rtfs of \( \pi' \) and \( \pi'' \) satisfy the following property:
\[
\forall t > |\pi'|, \quad \tau'.rf_{\pi'}(t) = \tau''.rf_{\pi''}(t) + C(\pi')
\]
where \( C(\pi') \) is a constant term that is only dependent on \( \pi' \) but not on \( t \)
\[
C(\pi') = \delta(x, m_1) + \gamma(y, m_t)
\]
Here \( m_1 = e(v_1) \) is the execution time of the vertex \( v_1 \) in \( \tau \), which is transformed to the set of vertices \( \nu'_{i,1}, \ldots, \nu'_{i,k_1,2} \) in \( \pi' \), \( m_1 \) and \( v_1 \) are defined similarly, and functions \( \delta(\cdot, \cdot) \) and \( \gamma(\cdot, \cdot) \) are defined as
\[
\delta(a, b) = \begin{cases} 
0 & \text{if } a = 1 \text{ or } a > b \\
 b - a + 1 & \text{if } 1 < a \leq b \\
 0 & \text{if } a > b 
\end{cases}
\]
\[
\gamma(a, b) = \begin{cases} 
 a - b & \text{if } 1 \leq a \leq b \\
 0 & \text{if } a > b 
\end{cases}
\]

**Proof.** We partition \( \pi' \) and \( \pi'' \) as follows

\[
\pi' = (\nu'_{1,x}, \ldots, \nu'_{1,k_1,2}, \nu'_{2,k_2,3}, \ldots, \nu'_{l-1,1}, \nu'_{l,1,\ldots,1_y})
\]
\[
\pi'' = (\nu''_{1,x}, \ldots, \nu''_{1,k_1,2}, \nu''_{2,k_2,3}, \ldots, \nu''_{l-1,1}, \nu''_{l,1,\ldots,1_y})
\]

(a) Consider \( \pi'_1 \) and \( \pi''_1 \): if \( x = 1 \), it means \( e(\pi'_1) = e(v_1) = e(\pi''_1) \); if \( 1 < x \leq m_1 \), \( e(\pi'_1) = \sum_{k=x}^{m_1} e(\nu'_{1,k}) = m_1 - x + 1 \), and \( e(\pi''_1) = 0 \); if \( x > m_1 \), \( e(\pi'_1) = e(\pi''_1) = 0 \). Summarizing the three cases,
\[
e(\pi'_1) = e(\pi''_1) + \delta(x, m_1)
\]
(b) Consider \( \pi'_{2,t-1} \) and \( \pi''_{2,t-1} \): we have
\[
e(\pi'_{2,t-1}) = \sum_{k=2}^{t-1} e(\nu_k) = e(\pi''_{2,t-1})
\]
(c) Consider \( \pi'_1 \) and \( \pi''_1 \): if \( 1 \leq y \leq m_1 \), then \( e(\pi'_1) = \sum_{k=1}^{y} e(\nu'_{1,k}) = y \), and \( e(\pi''_1) = m_1 \); if \( y > m_1 \), we have \( e(\pi'_1) = e(\pi''_1) = m_t \). Hence it holds
\[
e(\pi'_1) = e(\pi''_1) + \gamma(y, m_t)
\]
Combining the three subpaths, it yields
\[
\forall t > |\pi'|, \quad \tau'.rf_{\pi'}(t) - \tau''.rf_{\pi''}(t)
\]
\[
= e(\pi') - e(\pi'')
\]
\[
= (e(\pi'_1) - e(\pi''_1)) + (e(\pi'_{2,t-1}) - e(\pi''_{2,t-1})) + (e(\pi'_1) - e(\pi''_1))
\]
\[
= \delta(x, m_1) + 0 + \gamma(y, m_t)
\]
\[
= C(\pi')
\]
This lemma demonstrates that the rfs of isomorphic paths from $\tau'$ and $\tau''$ differ by a constant term which only depends on the corresponding start and end vertices. Thus, we introduce the definition of the refined request bound functions [28, 29] over a pair of vertices.

**Definition 10** [28, 29] Given a pair of vertices $v_i$ and $v_j$, the refined request bound function of a digraph task $\tau$ in the time interval $t$, denoted as $\tau.rbf(v_i, v_j, t)$, is defined as

$$\tau.rbf(v_i, v_j, t) = \max_{\pi: v_1 = v_i, v_l = v_j} \tau.rf_\pi(t)$$  \hspace{1cm} (29)

In other words, $\tau.rbf(v_i, v_j, t)$ is the maximum sum of execution times of any path $\pi = (v_1, \ldots, v_l)$ of $\tau$ such that

- the vertex corresponding to the first job is $v_1 = v_i$;
- the vertex corresponding to the last job is $v_l = v_j$;
- $p(\pi) < t$;
- $\tau.rbf(v_i, v_j, t) = \max \sum_{k=1}^{l} e(v_k) = e(\pi)$.

If $v_j$ is not reachable from $v_i$ within any time interval of length $t$, then we define $\tau.rbf(v_i, v_j, t) = -\infty$. With this definition, the domain for the possible rbf values is $\mathbb{R}^* = \mathbb{R} \cup \{-\infty\}$.

According to the definition, the rbf function of the task is the maximum among all the pairs $v_i$ and $v_j$ [28, 29]:

$$\tau.rbf(t) = \max_{i,j} \{\tau.rbf(v_i, v_j, t)\}$$  \hspace{1cm} (30)

We now derive a simple relationship between the refined request bound functions of $\tau'$ and $\tau''$.

**Lemma 13** For a generic digraph task $\tau$, consider a pair of vertices $\nu_{1,x}^i$ and $\nu_{1,y}^i$ in its transformed GDRT task $\tau'$ and the isomorphic pair of vertices $\nu_{1,x}^{i''}$ and $\nu_{1,y}^{i''}$ in its transformed UDRT task $\tau''$. We have the following property.

$$\forall t \geq 0, \quad \tau'.rbf(\nu_{1,x}^i, \nu_{1,y}^i, t) = \tau''.rbf(\nu_{1,x}^{i''}, \nu_{1,y}^{i''}, t) + \delta(x, m_1) + \gamma(y, m_l)$$  \hspace{1cm} (31)

where functions $\delta(\cdot, \cdot)$ and $\gamma(\cdot, \cdot)$ are defined in Equation (28).

**Proof.** Consider a path $\pi' = (\nu_{1,x}^i, \ldots, \nu_{1,y}^i)$ in $\tau'$ and its isomorphic path $\pi'' = (\nu_{1,x}^{i''}, \ldots, \nu_{1,y}^{i''})$ in $\tau''$. We first prove $\tau'.rbf(\nu_{1,x}^i, \nu_{1,y}^i, t)$ takes the maximal value with path $\pi'$ if and only if $\tau''.rbf(\nu_{1,x}^{i''}, \nu_{1,y}^{i''}, t)$ takes the maximal value with $\pi''$.

Let $\tilde{\pi}'$ in $\tau'$ denote any path whose start and end vertices are $\nu_{1,x}^i$ and $\nu_{1,y}^i$ with $p(\tilde{\pi}') < t$. $\tilde{\pi}''$ is the isomorphic path of $\tilde{\pi}'$ in $\tau''$. Obviously $p(\tilde{\pi}'') = p(\tilde{\pi}') < t$. 

Thus, we have
\[
\tau'.rbf(\nu_{1,x}',\nu_{1,y}',t) \text{ reaches its maximum with path } \pi'
\]
\[
\Rightarrow \forall \pi', \quad \tau'.rf_{\pi'}(t) \geq \tau'.rf_{\pi}(t)
\]
\[
\Rightarrow \forall \pi', \quad \tau'.rf_{\pi'}(t) - C(\pi') \geq \tau'.rf_{\pi}(t) - C(\pi)
\]
\[
\Rightarrow \forall \pi', \quad \tau'.rf_{\pi'}(t) - C(\pi') \geq \tau'.rf_{\pi}(t) - C(\pi')
\]
\[
\Rightarrow \forall \pi', \quad \tau'.rf_{\pi'}(t) \geq \tau'.rf_{\pi}(t)
\]
\[
(\text{by Lemma 12})
\]
\[
\Rightarrow \tau''.rbf(\nu_{1,x}'',\nu_{1,y}'',t) \text{ reaches its maximum with path } \pi''
\]
Since any pair of isomorphic paths \(\pi'\) and \(\pi''\) also satisfy Lemma 12, we conclude that the lemma holds.

We can derive the periodicity parameters of the \(rbf\)s of \(\tau'\) by the corresponding ones of the \(rbf\)s of \(\tau''\), shown in the following lemma.

**Lemma 14** For a generic digraph task \(\tau\) and its transformed GDRT task \(\tau'\) and UDRT task \(\tau''\), if there exist a real number \(q\) and a pair of integers \(x\) and \(p\) such that
\[
\forall t > r, \quad \tau''.rbf(t + p) = \tau''.rbf(t) + p \cdot q,
\]
we have
\[
\forall t > r, \quad \tau'.rbf(t + p) = \tau'.rbf(t) + p \cdot q.
\]

**Proof.** Because \(D(\tau)\) and consequently \(D''(\tau'')\) are strongly connected, for any two vertices \(\nu_{1,x}'\) and \(\nu_{1,y}'\) in \(D''(\tau'')\), it is:
\[
\forall t > r, \quad \tau''.rbf(\nu_{1,x}'',\nu_{1,y}'',t + p) = \tau''.rbf(\nu_{1,x}'',\nu_{1,y}'',t) + p \cdot q
\]
Thus, by Lemma 13, for any \(t > r\) we have
\[
\tau'.rbf(\nu_{1,x}',\nu_{1,y}',t + p) = \tau''.rbf(\nu_{1,x}'',\nu_{1,y}'',t + p) + \delta(x, m_1) + \gamma(y, m_1)
\]
\[
= \tau''.rbf(\nu_{1,x}'',\nu_{1,y}'',t) + p \cdot q + \delta(x, m_1) + \gamma(y, m_1)
\]
\[
= \tau'.rbf(\nu_{1,x}',\nu_{1,y}',t) + p \cdot q
\]
Since this is satisfied for any pair of vertices \(\nu_{1,x}'\) and \(\nu_{1,y}'\), by Equation (30) we have
\[
\forall t > r, \quad \tau'.rbf(t + p) = \tau'.rbf(t) + p \cdot q
\]

The following theorem shows that for any digraph, the linear period and linear factor of \(ibf\) are the same as those of \(rbf\), and the linear defect of \(ibf\) can be safely bounded by that of \(rbf\) plus one.

**Theorem 15** For a generic digraph task \(\tau\), if there exist a real number \(q\) and a pair of integers \(x\) and \(p\) such that
\[
\forall t > r, \quad \tau.rbf(t + p) = \tau.rbf(t) + p \cdot q,
\]we have
\[
\forall t > r + 1, \quad \tau.ibf(t + p) = \tau.ibf(t) + p \cdot q
\] (32)

This theorem is a direct result from Lemma 14 and Corollary 11, and we omit the proof here. With this result, the algorithms developed in [29] for calculating the linear periodicity parameters of \(rbf\) is reusable for \(ibf\).
5 Schedulability Analysis with Linear Upper Bound on $ibf$

We now describe the schedulability analysis that leverages the linear upper bound on $ibf$. This bound is intuitively tighter than that of $rbf$, as $ibf$ is a better approximation to the exact analysis than $rbf$. It is also illustrated in the following example.

Example 9 Figure 10 provides an illustrative example to describe the improvement of the linear upper bound on $ibf$ over that of $rbf$, using the task graph in Figure 1.

The following theorem formally described the linear upper bound on $ibf$.

**Theorem 16** For a DRT task $\tau$ containing cycles,

$$\forall t \geq 0, \quad \tau.ibf(t) \leq C^{ibf} + t \cdot \lambda(\tau)$$

where

$$C^{ibf} = \max_{\pi \in S(\tau)} \left\{ e(\pi) - p^\Sigma(\pi) \cdot \lambda(\tau) \right\}$$

$S(\tau)$ represents the set of all elementary paths in $D(\tau)$, and $p^\Sigma(\pi)$ for the path $\pi = (v_1, \ldots, v_l)$ is defined as

$$p^\Sigma(\pi) = p(\pi) + e(v_l) = \sum_{i=1}^{l-1} p(v_i, v_{i+1}) + e(v_l)$$

**Proof.** We first prove for any path $\pi = (v_1, \ldots, v_l)$ it holds

$$\tau.ibf(t_0) = e(\pi) \leq C^{ibf} + t_0 \cdot \lambda(\tau)$$

where $t_0 = p^\Sigma(\pi) = p(\pi) + e(v_l)$. 
(a) If \( \pi \in S(\tau) \), i.e., it is an elementary path, then we have
\[
C^{of} + t_0 \cdot \lambda(\tau) \geq e(\pi) - t_0 \cdot \lambda(\tau) + t_0 \cdot \lambda(\tau) = \tau . \text{if}_{\pi}(t_0)
\]
(b) Otherwise, there must exist a cycle in \( \pi \). With the same construction as the proof of Lemma V.2 in [20], \( \pi \) can be represented as one combination of an elementary path \( \pi^* \) and some cycles \( \pi^{(j)} = (v_1^{(j)}, \cdots, v_i^{(j)}) \) where \( v_i = v_1^{(j)} = v_i^{(j)} \). It can be verified that
\[
\begin{aligned}
e(\pi) &= e(\pi^*) + \sum_j \sum_{i=1}^{l_j-1} e(v_i^{(j)}) \\
p^\Sigma(\pi) &= p^\Sigma(\pi^*) + \sum_j \sum_{i=1}^{l_j-1} p(v_i^{(j)}, v_{i+1}^{(j)}). 
\end{aligned}
\tag{36}
\]
Clearly, \( e(\pi^*) \leq C^{of} + p^\Sigma(\pi^*) \cdot \lambda(\tau) \) since \( \pi^* \) is an elementary path. Moreover, since \( \lambda(\tau) \) is the maximal mean over all cycles, for all cycles \( \pi^{(j)} \) it holds
\[
\sum_{i=1}^{l_j-1} e(v_i^{(j)}) = p(\pi^{(j)}) \cdot \lambda(\pi^{(j)}) \leq p(\pi^{(j)}) \cdot \lambda(\tau)
\]
The summation of all cycles of \( \pi \) can be bounded by
\[
\sum_j \sum_{i=1}^{l_j-1} e(v_i^{(j)}) \leq \sum_j p(\pi^{(j)}) \cdot \lambda(\tau).
\]
Finally, we apply this to Equation (36) and get
\[
e(\pi) \leq C^{of} + p^\Sigma(\pi^*) \cdot \lambda(\tau) + \sum_j p(\pi^{(j)}) \cdot \lambda(\tau)
= C^{of} + p^\Sigma(\pi) \cdot \lambda(\tau)
\]
By the above analysis, it holds \( \tau . \text{if}_{\pi}(t_0) \leq C^{of} + t_0 \cdot \lambda(\tau) \).

Now we prove that for any \( t \geq 0 \), \( \tau . \text{if}_{\pi}(t) \leq C^{of} + t \cdot \lambda(\tau) \). We discuss according to the position of point \( (t, \tau . \text{if}_{\pi}(t)) \):

**Case 1:** \((t, \tau . \text{if}_{\pi}(t))\) is at a slanted segment of \( \tau . \text{if}_{\pi} \) (e.g., \( t = 1 \) in Figure 2). Assuming \( t_0 = \min\{t' | t' \geq t \land t' = p^\Sigma(\pi') \} \) where \( \pi' \) is a prefix of \( \pi \), i.e., \( t_0 \) is the ending point of the slanted segment containing \( t \). We have
\[
\tau . \text{if}_{\pi}(t) = \tau . \text{if}_{\pi}(t_0) - (t_0 - t)
\leq C^{of} + t_0 \cdot \lambda(\tau) - (t_0 - t) \quad \text{(by Equation (35))}
= C^{of} + t \cdot \lambda(\tau)
\]

**Case 2:** \((t, \tau . \text{if}_{\pi}(t))\) is at a horizontal segment of \( \tau . \text{if}_{\pi} \) (e.g., \( t = 3 \) in Figure 2). Assuming \( t_0 = \max\{t' | t' \leq t \land t' = p^\Sigma(\pi') \} \) where \( \pi' \) is a prefix of \( \pi \), i.e., \( t_0 \) is the starting point of the horizontal segment containing \( t \). We have
\[
\tau . \text{if}_{\pi}(t) = \tau . \text{if}_{\pi}(t_0)
\leq C^{of} + t_0 \cdot \lambda(\tau)
\leq C^{of} + t \cdot \lambda(\tau)
\]
Combining the two cases, it implies that $\tau \cdot \text{ibf}(t) \leq C^{ibf} + t \cdot \lambda(\tau)$. Further, as $\pi$ is an arbitrary path in the $D(\tau)$, we have $\tau \cdot \text{ibf}(t) \leq C^{ibf} + t \cdot \lambda(\tau)$. $\square$

In [28, 29], a linear upper bound on $rbf$ is derived as

$$\forall t \geq 0, \tau \cdot rbf(t) \leq C^{rbf} + t \cdot \lambda(\tau)$$

which means that the linear upper bounds of $rbf$ and $ibf$ have the same slope $\lambda(\tau)$ for any task $\tau$. The calculation of $C^{rbf}$, the intercept of the linear bound on $rbf$, uses the transformed UDRT [28, 29]. Here, we give its equivalent form by directly operating on the original digraph like Equation (34):

$$C^{rbf} = \max_{\pi \in S(\tau)} \{e(\pi) - p(\pi) \cdot \lambda(\tau)\}$$  \hspace{1cm} (37)

where $S(\tau)$ represents the set of all elementary paths in $D(\tau)$. The above equation can be proven similarly as Theorem 16, i.e., for any path $\pi$, $\tau \cdot rbf(t_0) = e(\pi) \leq C^{rbf} + t_0 \cdot \lambda(\tau)$ where $t_0 = p(\pi)$. The following theorem shows that the linear upper bound on $ibf$ is always no larger than that on $rbf$.

**Theorem 17** For any digraph task $\tau$ containing cycles, we have $C^{ibf} \leq C^{rbf}$.

**Proof.** For any path $\pi$, it always holds $p(\pi) \leq p(\pi) + e(v_1) = p^{\Sigma}(\pi)$ where $v_1$ is the last vertex in $\pi$. Hence $e(\pi) - p^{\Sigma}(\pi) \cdot \lambda(\tau) \leq e(\pi) - p(\pi) \cdot \lambda(\tau)$. Since this is satisfied for any path $\pi$, it implies $C^{ibf} \leq C^{rbf}$. $\square$

For a vertex $v$ and the interference task set $\Gamma$, we can provide a closed form formula for the response time bounds by using $ibf$ under arbitrary deadline in the following theorem.

**Theorem 18** Consider a vertex $v$ in task $\tau_v$, whose interference tasks set is $\Gamma$. If $\lambda^*(\tau_v) + \sum_{\tau_i \in \Gamma} \lambda(\tau_i) \leq 1$, the upper bound on the worst case response time for $v$ can be calculated by

$$R_{LU IBF}(v, \Gamma) = \frac{e(v) + \sum_{\tau_i \in \Gamma} \tau_i \cdot C^{ibf}}{1 - \sum_{\tau_i \in \Gamma} \lambda(\tau_i)}$$ \hspace{1cm} (38)
Proof. Leveraging Equation (20), we have

\[
R_{IBF}(v, \Gamma) = \max_{\pi_e \in D(v)} \left\{ \min_{t \geq 0} \left\{ t - p(\pi_e) \left[ \tau_{e, rf}(t) + \sum_{\tau_i \in F} \tau_i, ibf(t) \leq t \right] \right\} \right\}
\]

\[
\leq \max_{\pi_e \in D(v)} \left\{ \min_{t \geq 0} \left\{ t - p(\pi_e) \left[ \tau_{e, rf}(t) + \sum_{\tau_i \in F} \tau_i, C^{ibf} + t \cdot \lambda(\tau_i) \right] \leq t \right\} \right\}
\]

\[
\leq \max_{\pi_e \in D(v)} \left\{ e(\pi_e) + \sum_{\tau_i \in F} \frac{1}{1 - \sum_{\tau_i \in F} \lambda(\tau_i)} \lambda(\tau_i) - p(\pi_e) \right\}
\]

\[
\leq \max_{\pi_e \in D(v)} \left\{ e(\pi_e) + \sum_{\tau_i \in F} \frac{1}{1 - \sum_{\tau_i \in F} \lambda(\tau_i)} \lambda(\tau_i) - p(\pi_e) \right\}
\]

Since \( \lambda^*(\tau_v) + \sum_{\tau_i \in F} \lambda(\tau_i) \leq 1 \), it means for any path \( \pi_v \) of \( \tau_v \), \( \frac{\lambda(\tau_v)}{1 - \sum_{\tau_i \in F} \lambda(\tau_i)} - 1 \leq 0 \).

Thus, the term \( p(\pi_v) \cdot \left\{ \frac{\lambda(\pi_v)}{1 - \sum_{\tau_i \in F} \lambda(\tau_i)} - 1 \right\} \) has a maximal value 0 (e.g., when \( p(\pi_v) = 0 \) where \( \pi_v \) only has one vertex \( v \)). Hence, we have

\[
R_{IBF}(v, \Gamma) \leq \frac{e(\pi_v) + \sum_{\tau_i \in F} \tau_i, C^{ibf}}{1 - \sum_{\tau_i \in F} \lambda(\tau_i)}
\]

\[
\square
\]

Example 10 The example in Figure 5 also shows that in Theorem 18 the assumption \( \lambda^*(\tau_v) + \sum_{\tau_i \in F} \lambda(\tau_i) \leq 1 \) is necessary (otherwise the response time bound is not safe).

There are two digraph tasks where \( \lambda^*(\tau_2) = 0.75 \) and \( \lambda(\tau_1) = 0.5 \). Hence the above assumption is violated. By Equation (34), \( \tau_1, C^{ibf} = 0.5 \). Hence by Equation (38)

\[
R_{LUBIBF}(v_3, \Gamma) = \frac{e(v_3) + \tau_1, C^{ibf}}{1 - \lambda(\tau_1)} = \frac{1 + 0.5}{1 - 0.5} = 3
\]

However, \( R(v_3, \Gamma) = 3.5 \) as calculated in Example 5. This demonstrates that Equation (38) cannot safely bound the response time if the assumption is violated.
Theorem 18 is a generalization of the response time bound in [7]. More specifically, for a sporadic task $\tau$ with period $T$ and WCET $C$, Bini et al. [7] develop a linear upper bound on the $ibf$ function (it was called the worst case workload) of $\tau$ as follows

$$\forall \tau, \quad \tau_ibf(t) \leq C \cdot (1 - \lambda(\tau)) + t \cdot \lambda(\tau)$$  \hspace{1cm} (39)

where $\lambda(\tau) = \frac{C}{T}$ denotes the utilization of the task. The upper bound on the response time of a task $\tau$ with a set of interfering task $I'$ is given as

$$R_{LU_BIBF}(\tau, I') = \frac{C + \sum_{\tau_i \in I'} C_i(1 - \lambda(\tau_i))}{1 - \sum_{\tau_i \in I'} \lambda(\tau_i)}$$ \hspace{1cm} (40)

On the other hand, the sporadic task model is a special case of the digraph task model: $\tau$ can be modeled as a digraph with a single vertex $v$ and a self-loop edge $(v, v)$, where $e(v) = C$ and $p(v, v) = T$. The only elementary path in $\tau$ is $\pi = (v)$. By Theorem 16, $p^\Sigma(\pi) = p(\pi) + e(v) = C$, and

$$C^{ibf} = e(\pi) - p^\Sigma(\pi) \cdot \lambda(\tau) = C(1 - \lambda(\tau))$$

By Theorem 18, the response time upper bound on $\tau$ is

$$R_{LU_BIBF}(\tau, I') = \frac{C + \sum_{\tau_i \in I'} \tau_i.C^{ibf}}{1 - \sum_{\tau_i \in I'} \lambda(\tau_i)} = \frac{C + \sum_{\tau_i \in I'} C_i(1 - \lambda(\tau_i))}{1 - \sum_{\tau_i \in I'} \lambda(\tau_i)}$$

The above two equations are precisely the same as Equations (39) and (40), the results from [7].

The upper bound on the worst case response time can be used as a sufficient condition for checking schedulability: if $R_{LU_BIBF}(v, I') \leq d(v)$, then the vertex $v$ of $\tau_i$ is schedulable, otherwise its schedulability is uncertain (which might require more precise analysis). The calculation of $R_{LU_BIBF}(v, I')$ provides an approach to quickly check schedulability. Similarly, we can derive another response time upper bound by replacing $C^{ibf}$ with $C^{rbf}$:

$$R_{LU_BRBF}(v, I') = \frac{e(v) + \sum_{\tau_i \in I'} \tau_i.C^{rbf}}{1 - \sum_{\tau_i \in I'} \lambda(\tau_i)}$$ \hspace{1cm} (41)

By Theorem 17, it always holds that $R_{LU_BIBF}(v, I') \leq R_{LU_BRBF}(v, I')$.

We now study the quality of the upper bounds on response times by Equation (38) and (41). First, $R_{LU_BIBF}$ ($R_{LU_BRBF}$) is a looser estimation the response time than $R_{IBF}$ ($R_{RBF}$). Since $R_{IBF}$ and $R_{RBF}$ have no approximation ratio, by Theorem 5, $R_{LU_BIBF}$ and $R_{LU_BRBF}$ also have no approximation ratio. As for the speedup factor, we have the following theorem.

**Theorem 19** $R_{LU_BIBF}$ and $R_{LU_BRBF}$ have no speedup factor.
**Proof.** For any given speedup factor $u' > 1$, we find a counterexample for $u = \lceil u' \rceil$ as follows. We consider a task $v_0$ containing only one vertex $v$ with $e(v) = 1/k$ where $k > 2 + \frac{1}{u'}$. The higher priority task set is $I = \{\tau\}$ where $\tau$ is illustrated in Figure 11. The notations $\lambda^u(\tau)$, $C^{ibf,u}$, $C^{rbf,u}$, $R^u_{LUBIBF}$, and $R^u_{LUBRBF}$ are defined as the counterparts on a speed-$u$ ($u > 1$) processor.

We note that $C^{ibf,u}$ and $C^{rbf,u}$ take the maximum when $\pi = (v_1, v_2, \ldots, v_{u^2})$. Thus, we can verify that $\lambda^u(\tau) = \frac{1}{uk}$, $e(\pi) = u^2$, and $p(\pi) = 2(u^2 - 1)$. By Equations (34) and (37)

$$
\begin{aligned}
C^{ibf,u} &= e(\pi)/u - (p(\pi) + e(v_{u^2}))/u \cdot \lambda^u(\tau) = u - \frac{2u^2 - 2 + 1/u}{uk} \\
C^{rbf,u} &= e(\pi)/u - p(\pi) \cdot \lambda^u(\tau) = u - \frac{2u^2 - 2}{uk}
\end{aligned}
$$

Applying this to Equations (38) and (41), we get the upper bounds on the response time as follows:

$$
\begin{aligned}
R^u_{LUBIBF}(v, I) &= \frac{e(v)/u + C^{ibf,u}}{1 - \lambda^u(\tau)} \\
&= \frac{1}{uk} + u - \frac{2u^2 - 2 + 1/u}{uk} \\
&= u^2k + 3 - 1/u - 2u^2 \\
&= \frac{uk - 1}{uk} \\
R^u_{LUBRBF}(v, I) &= \frac{e(v)/u + C^{rbf,u}}{1 - \lambda^u(\tau)} \\
&= \frac{1}{uk} + u - \frac{2u^2 - 2}{uk} \\
&= \frac{u^2k + 3 - 2u^2}{uk - 1}
\end{aligned}
$$

It is easy to verify that the exact worst case response time is $R(v, I) = 1 + 1/k$. Because $u > 1$ and $k > 2 + \frac{3}{u-1}$, we have $uk - 2u - k - 1 > 0$ and $uk - 1 > 0$. Consequently,

$$
\begin{aligned}
R^u_{LUBIBF}(v, I) - R(v, I) &= \frac{u(uk - 2u - k - 1) + 4 - 1/u + 1/k}{uk - 1} > 0 \\
R^u_{LUBRBF}(v, I) - R(v, I) &= \frac{u(uk - 2u - k - 1) + 4 + 1/k}{uk - 1} > 0
\end{aligned}
$$

**Fig. 11** An example digraph task $\tau$ for Theorem 19.
That is, for any $u' > 1$, we can construct a counterexample for $u = \lceil u' \rceil$ such that $R_{ULUBIBF}(v, \Gamma) > R(v, \Gamma)$ and $R_{ULUBRBF}(v, \Gamma) > R(v, \Gamma)$. Hence, there is no speedup factor for either of the two linear upper bounds.

$C^{ibf}$ can also be used to get an improved upper bound on $t_f$ compared to Equation (15) for tasks with l-MAD property.

$$
 t_f = \frac{\tau_{C^{dbf}} + \sum_{\tau_i \in F} \tau_i C^{ibf}}{1 - \lambda(\tau) - \sum_{\tau_i \in F} \lambda(\tau_i)}
$$

(42)

The above equation can be derived in the similar way as Equation (25) in [29] and we omit the proof here.

We now summarize the relationships among the response times from different analysis methods, where $R_A \leq R_B$ means that the response time from method $A$ is no larger than that from method $B$ (hence $A$ is more accurate than $B$).

$$
 R_{RF} \leq R_{RBF} \leq R_{ULUBIBF} \leq R_{ULUBRBF}
$$

(43)

In the above equation, the only pair of analysis results for which we cannot draw a conclusive relationship is $R_{RBF}$ and $R_{ULUBIBF}$, meaning neither is more accurate than the other all the times. This is also demonstrated in the experiments of Section 6.2.1.

6 Experimental Evaluation

In this section, we evaluate the efficiency improvement on the schedulability analysis due to the periodicity of $ibf$. We also study the quality of the four approximate response times: $R_{ULUBRBF}$, $R_{ULUBIBF}$, $R_{RBF}$, and $R_{IBF}$. These experiments are implemented in the C++ language and executed on a machine with an Intel Core i7 3.4GHz CPU.

6.1 Analysis Efficiency Improvement from Linear Periodicity

In the first experiment, we evaluate the improvement on the schedulability analysis with the periodicity property. Our experiment configuration is similar to [28, 29], but with some minor modifications. We generate random task systems consisting of $n = 5$ to 40 digraph tasks, where each digraph has 1 – 40 nodes. The number of outgoing edges from a node is randomly distributed as follows: 40% of the node has one outgoing edge, 40% with two, 10% with three, and 10% with four. These outgoing edges are randomly connected to any other node (including the source itself). The average degree of nodes is around 3.6 and 54.8% of the digraph tasks are strongly connected. The base period of each task is generated by the product of one to three factors, each randomly drawn from the harmonic sets (2, 4), (6, 12),
The inter-release time is scaled by a factor randomly extracted from the set \{1, 2, 4, 5, 10\}. The priorities of the digraph tasks are assigned using the rate monotonic policy based on their base periods. The deadline of vertex \(v\) is set to be
\[
ed(v) = \min_{(v,u) \in E} \text{random}(0,2) \times \max_{(v,u) \in E} p(v, u),
\]
where function \text{random}(0,2) returns a random real number between 0 and 2. Thus, the task system has arbitrary deadlines. The system utilization \(U\) ranges from 50\% to 95\% and each task utilization is calculated by the UUniFast algorithm [6]. The execution times are randomly selected to satisfy the utilization. For each data we generated 1000 random systems. We focus on the schedulability of the lowest-priority task for each system. The schedulability test is based on that of \(ibf\) in Equation (20).

As part of the schedulability analysis, \(ibf\) of each higher priority task shall be computed. We compare our method with the one in [11] that does not study the periodicity. Our approach needs to additionally calculate the linear factor \(q\), the linear period \(p\) and the linear defect \(r\) on the \(ibf\). Figure 12 shows the average runtime of the schedulability analysis versus system utilization (\(U\)) for systems with 20 tasks. It also illustrates the improvement ratio defined as
\[
\frac{\text{Analysis Runtime without periodicity}}{\text{Analysis Runtime with periodicity}} - 1.
\]
Note that a positive improvement ratio means that we can speedup the schedulability analysis using the periodicity property. Figure 13 depicts the improvement ratio versus the number of tasks \(n\) under several system utilization settings \(U = 60\%, 70\%, 80\%\) and \(90\%\). These figures indicate that the periodicity property can significantly reduce the
Fig. 13 The runtime improvement versus the number of tasks.

analysis complexity. For example, the improvement ratio is 151\% when \( n = 10 \) and \( U = 80\% \).

6.2 Evaluation of Approximate Response Times

In this experiment, we evaluate the accuracy of the upper bounds on response times \( R_{LUBRB} \) and \( R_{LUBIB} \) calculated by Equations (38) and (41) respectively. We use the approximate response times \( R_{RBF} \) and \( R_{IBF} \) computed in Equations (19) and (20) respectively.

6.2.1 Effect of Task Periods with Two-task Systems

We first study the impacts of the digraph task periods on the quality of the analysis methods. We follow the configuration in [7], and generate task systems that only consist of exactly two tasks. The higher priority task is generated like those in the first experiment (Section 6.1), but with a fixed number of nodes \( k = 2, 5, 10, \) and \( 20 \). We denote the linear period of the higher priority digraph task \( T_h \) as \( T_h \). Meanwhile, the lower priority task \( T_l \) is periodic with period \( T_l \). The ratio \( T_h/T_l \) ranges from 0 to 1. The utilizations of the two tasks satisfy \( U_h/U_l = 0.25 \) and \( U = U_h + U_l \in \{0.2, 0.4, 0.6, 0.8\} \). We generate 1000 such test systems for each \( U \) and \( k \) configuration, and calculate the approximate response times of task \( T_l \).
Fig. 14 Approximate response times of $\tau_i$ (number of nodes in $\tau_h$ is $k = 2$).
Fig. 15 Approximate response times of $\tau_l$ (number of nodes in $\tau_h$ is $k = 5$).
Fig. 16 Approximate response times of $\tau_i$ (number of nodes in $\tau_i$ is $k = 10$).
Fig. 17 Approximate response times of $\tau_i$ (number of nodes in $\tau_h$ is $k = 20$).
Figures 14–17 show the curves of the response times normalized by $T_l$ under different configurations of $U$ and $k$. There are a few observations from the figures. First, $R_{LUBRBF}$ and $R_{LUBIBF}$ are close to $R_{IBF}$ and $R_{RBF}$ when the ratio $T_h/T_l$ approaches zero. This means that when $T_l$ is significantly larger than $T_h$, the response time upper bounds $R_{LUBRBF}$ and $R_{LUBIBF}$ introduce very small pessimism. Second, $R_{LUBRBF}$ and $R_{LUBIBF}$ are discontinuous at some specific ratio $T_h/T_l$, which is because an additional job of $\tau_h$ interferes with $\tau_l$. Third, in general, the gap between $R_{LUBIBF}$ and $R_{IBF}$ becomes larger with higher system utilization. This observation also applies to $R_{LUBRBF}$ and $R_{RBF}$. Finally, the results confirm the relationship in Equation (43) that $R_{LUBIBF}$ always has a higher accuracy than $R_{LUBRBF}$, and $R_{IBF}$ is always tighter than $R_{RBF}$. As for $R_{LUBIBF}$ and $R_{RBF}$, no one is always better than the other: in most of the cases $R_{RBF}$ is more accurate, but at some cases (e.g., in Figure 17(c) and 17(d)) it is the opposite.

6.2.2 Effect of Task System Parameters with Larger Systems

Finally, we evaluate these analysis methods with larger task systems. Each test system contains $n+1$ tasks: $\tau_l$ is the task under analysis that is periodic with the lowest priority; $n$ higher priority digraph tasks (denoted as $\tau_i$, $i = 1, \ldots, n$). $\tau_l$ always has a utilization of 10% and the total system utilization $U = \sum_{i=1}^{n} U_i + U_l \in \{0.4, 0.6, 0.8\}$ (representing light, moderate, and heavy workload respectively). The utilization of each higher priority task is randomly distributed by the UUniFast algorithm [6]. We then follow the same configuration as the first experiment (Section 6.1) to generate the $n$ digraph tasks. Finally, the period of $\tau_l$ is set to $T_l = T_n/x$ where $T_n$ is the linear period of $\tau_n$ (the largest among the $n$ digraph tasks), and factor $x$ is drawn from the set $\{0.2, 0.5, 0.8\}$.

Figures 18–20 illustrate the average results of 1000 systems at each configuration. For convenience, the response times are also normalized by $T_l$. We observe that the ratio $R_l/T_l$ decreases with increasing number of tasks $n$ or decreasing system utilization $U$. In the figures, the response time bounds $R_{LUBIBF}$ and $R_{LUBRBF}$ are quite close to $R_{IBF}$ and $R_{RBF}$, and the difference becomes smaller with lower utilization $U$, larger task number $n$, or smaller factor $x$.

7 Conclusion

In this work, we focus on improving the schedulability analysis of digraph tasks with static priorities. We generalize the analysis to systems with arbitrary deadlines. We demonstrate the periodicity of interference bound functions so that the complexity of calculating $ibf$ of large time intervals with length $t$ is independent from $t$. Moreover, a tight linear upper bound on $ibf$ is developed to further speed up the schedulability analysis, which can be used to provide a tighter response time upper bound. We use a series of experiments to demonstrate the improvement on analysis accuracy and runtime from the proposed methods.
Fig. 18 Approximate response times of $\tau_1$ (ratio $T_u/T_i = 0.2$).
Fig. 19 Approximate response times of \( \tau_t \) (ratio \( T_u/T_t = 0.5 \)).
Fig. 20 Approximate response times of $\tau_l$ (ratio $T_u/T_l = 0.8$).
References