Parameter Adaption for Generalized Multiframe Tasks and Applications to Self-Suspending Tasks

Bo Peng and Nathan Fisher

Department of Computer Science, Wayne State University, USA. Email: \{et7889,fishern\}@wayne.edu

Abstract—The generalized multiframe task model (GMF) extends the sporadic task model and multiframe task model. Each frame in the GMF model contains an execution time, a relative deadline, and a minimum inter-arrival time. These parameters are fixed after task specification time in the GMF model. However, multimedia and adaptive control systems may be overloaded and no longer stabilized when the task parameters in such systems are not flexible. In order to address this problem, deadlines and periods may change to alleviate temporal overload, for example in the parameter adaption and elastic scheduling model. In this paper, we propose a new model GMF-PA (the GMF model with parameter adaption). This model allows task parameters to be flexible in arbitrary-deadline systems. A necessary schedulability test based on mixed-integer linear programming (MILP) is given to check the schedulability under EDF scheduling and optimally assign deadlines and periods at the same time. We also prove that the test is a sufficient and necessary schedulability test when task parameters must be integers. An approximation algorithm is also deployed to reduce computational running time. The speed-up factor of our approximation algorithm is $1 + \epsilon$ where $\epsilon$ can be arbitrarily small, with respect to the exact schedulability test of GMF-PA tasks under EDF. We also apply the GMF model to self-suspending tasks. By extending recent work on scheduling self-suspending tasks, we remove the assumption that deadlines are equally assigned in self-suspending tasks, and the system is extended from constrained-deadline systems to arbitrary-deadline systems. We have done exhaustive experiments to show that the schedulability ratio is improved using our techniques in our GMF-PA model.

Index Terms—generalized multiframe task model; multiframe self-suspending tasks; uniprocessor scheduling; mixed-integer linear programming; approximation algorithms.

I. INTRODUCTION

In real-time systems, worst-case execution time (WCET) analysis calculates an upper bound for each task based on the total aggregate amount of execution required for a job. Such estimates derived from WCET analysis are used in real-time schedulability analysis to determine whether every job in a system can finish executing before its deadline. Therefore, the effectiveness of the schedulability analysis hinges upon the precision of WCET estimates. Unfortunately, many scheduler properties that simplify schedulability analysis often introduce pessimism into WCET analysis. For example, the oft-assumed property of a task is that the worst-case execution times are the same for all jobs. However, this assumption is inaccurate for tasks which produce a sequence of jobs with heterogeneous execution times. For example in multimedia systems, the execution time of a job containing video-data is not necessarily the same with the execution time of the next job containing corresponding audio-data.

The multiframe task model [15] (MF) generalizes the periodic task model and reduces the pessimism by a set of finite recurring frames. The finite set of frames can be seen as a cycle. This cycle recurs an infinite number of times. Frames can have different WCETs instead of identical ones. The generalized multiframe task model [4] (GMF) further generalizes the sporadic task model and multiframe task model [15]. Instead of setting implicit deadlines and same minimal inter-arrival times for each frame in the MF model, the GMF model assigns an individual deadline and a minimal inter-arrival time for each frame.

The GMF model increases flexibility compared to the sporadic task model and multiframe task model. Even with this, however, the parameters in the GMF model are fixed during task specification time. The schedulability will be decreased when the parameters are not flexible and dynamic, e.g., in multimedia and adaptive control systems. Buttazzo et al. [7] defined the elastic model in which each period has an upper bound. If a job misses a deadline, the period is allowed to increase under the upper bound. Chantem et al. [8] selected the deadlines and periods in the generalized elastic model. The elastic model is allowed to change parameters during runtime. In this paper, we extend the GMF model to let the periods and deadlines be selected under a set of upper bounds. In this flexible model, the period and deadline are optimally assigned prior to runtime for each frame under our methods.

We also apply this flexible GMF model to schedule a set of self-suspending tasks under EDF scheduling. In self-suspending tasks, a self-suspension delay occurs when a job invokes some external operations [14] such as requests for computation offloading, I/O operations, etc. In order to address the pessimistic performance when considering the self-suspension delay as a part of execution time, recent work [9], [13] presented fixed-relative-deadline (FRD) scheduling algorithm. FRD scheduling breaks a task into computation phases and suspensions phases; each computation phase can be viewed as a frame with its own relative deadline and execution. Under FRD scheduling, a simple deadline assignment approach is presented that equally assigns deadlines [9], [13] based on the difference between the self-suspension and minimum inter-arrival time. However, the proposed deadline

This research has been supported in part by an NSF CAREER Grant (No. CNS-0953585) and a grant from Wayne State University’s Office of Vice President of Research.
assignment is very restrictive, and we will detail the restriction in the beginning of Section VI. We consider a more general, less restrictive deadline assignment strategy to improve schedulability for multi-segment self-suspending tasks. The system is also extended to an arbitrary-deadline system.

**Our Contributions:**

- We propose a new model GMF-PA (the generalized multiframe model with parameter adaption) which permits a flexible selection of the relative deadline and minimum separation time for each frame under EDF scheduling.
- We develop a parameter-adaption algorithm that selects the deadlines and minimum inter-arrival times in the GMF-PA model. We prove that the algorithm is a necessary schedulability test. When parameters are assumed to be integers, we also prove that the algorithm is a sufficient and necessary schedulability test, i.e., an exact test.
- We develop an approximation algorithm and prove that this method is a sufficient schedulability test.
- We apply our parameter adaption algorithm and its approximation algorithm on multiple-segment self-suspending tasks. The system is also extended to an arbitrary-deadline system.
- The speed-up factor of our approximation algorithm is $1 + \epsilon$ with respect to the exact schedulability test of GMF-PA tasks under EDF.
- We implement exhaustive experiments to show the improvement compared to previous work.

We introduce our GMF-PA model in Section II. Section III surveys the related work. Section IV presents our parameter-adaption method by using mixed-integer linear programming (MILP) to get a necessary schedulability test under EDF. Since MILP is not scalable in general, an approximation algorithm of MILP is presented in Section V. Section VI applies the parameter-adaption method and its approximation algorithm to self-suspending tasks. Section VII does exhaustive experiments compared with state-of-art results. At last, Section VIII concludes this work and proposes future work.

**II. MODEL**

We define our GMF-PA model based on the GMF model in Section II-A. The multiple-segment self-suspending task model and how to apply our GMF-PA model to the multiple-segment self-suspending tasks are presented in Section II-B.

**A. The Generalized Multiframe Model with Parameter Adaptation**

In this section, we introduce the generalized multiframe model (GMF) and define our generalized multiframe model with parameter adaptation (GMF-PA).

In the GMF model [4], a task $t_i$ is represented by the 3-tuple $(E_i, D_i, P_i)$. The vector $E_i = [E_i^0, E_i^1, \ldots, E_i^{N_i-1}]$ represents execution times, $D_i = [D_i^0, D_i^1, \ldots, D_i^{N_i-1}]$ represents relative deadlines, and $P_i = [P_i^0, P_i^1, \ldots, P_i^{N_i-1}]$ represents minimum inter-arrival times between consecutive frames. Each frame $t_i^j$ is characterized by an arrival time $a_i^j$, a deadline $a_i^j + d_i^j$, and a worst-case execution time $e_i^j$ as:

![Figure 1](image-url)
period $P_i$ is the upper bound of $\sum_{j=0}^{N_i-1} P_i^j$. The utilization of task $\tau_i$ is $U_i = E_i/P_i$, and the utilization of a task system is $U_{\text{cap}} = \sum_{i=0}^{n-1} U_i$.

We aim to optimally select relative deadlines ($D_i^j$) and minimum inter-arrival times ($P_i^j$) in this paper, under the basic requirements as follows:

1) $E_i^j \leq D_i^j \leq D_i^j$, \forall $i, j$
2) $E_i^j \leq P_i^j \leq P_i^j$, \forall $i, j$
3) $D_i^j \leq P_i^j + D_i^{j+1}$\mod $N_i$, \forall $i, j$

4) $\sum_{j=0}^{N_i-1} D_i^j \leq D_i$, \forall $i$
5) $\sum_{j=0}^{N_i-1} P_i^j \leq P_i$, \forall $i$

Figure 1 is also an example of a GMF-PA model as long as the parameters satisfy the basic requirements. A task system must obey the first two inequalities to be feasible. The third inequality is required by the $l$-MAD property. The last two inequalities check whether a system is feasible under the upper bounds $D_i$ and $P_i$. We call the first three constraints as frame constraints and the last two constraints as cycle constraints in the rest of this paper.

B. The Multiple-segment Self-suspending Task Model and the GMF-PA Model

Scheduling tasks with self-suspending has received renewed interests in recent years. A task suspends itself to communicate with external devices, I/O operations, computation offloading, etc. A typical model derived from such systems contains two computation frames separated by a self-suspending frame. After the first computational frame finishes, the job suspends executing the other computational frame until such an external operation completes. The multiple-segment self-suspending task model [13] allows a task to suspend many times. Huang and Chen [13] where the first to identify the relationship between self-suspending tasks and the GMF model call a computational frame as a computation segment and a suspending frame as a suspension interval; we call both frames instead to be congruent with the GMF-PA model.

Figure 2 shows an example of a multiple-segment self-suspending task model.

In the multiple-segment self-suspending task model [13], the task system $T = \{\tau_0, \tau_2, \ldots, \tau_{n-1}\}$ is the task system which consists of $n$ sporadic self-suspending tasks executing on one processor. Each task $\tau_k = ((E_i^j, D_i^j, S_i^j, \ldots, S_i^{m_i-2}, E_i^{m_i-1}), P_i, D_i)$ has $m_i$ computational frames and $m_i - 1$ suspending frames. The execution time of a computational frame is $E_i^j$ and the self-suspending delay of a suspending frame is $S_i^j$. The frames work in order and $j + 1$th frame cannot start executing until the time when $j$th frame finishes executing. The period is $P_i$ and relative end-to-end deadline is $D_i$.

In order to incorporate with our GMF-PA model, each frame is rewritten as $\phi_i^j = (E_i^j, D_i^j, S_i^j, \ldots, E_i^{m_i-1}, S_i^{m_i-2}, \ldots, E_i^{m_i-1})$. The total number of frames in a cycle of $\tau_i$ is $N_i = 2 \cdot m_i - 1$. For a computational frame, $E_i^j$ and the parameters in frame constraints consist of a frame $\phi_i^j = (E_i^j, D_i^j, S_i^j, \ldots, E_i^{m_i-1}, S_i^{m_i-2}, \ldots, E_i^{m_i-1})$. For a suspending frame $\phi_i^k$, $E_i^k = 0$, $D_i^k = D_i^k = E_i^k = S_i^k$. We change the superscripts of frames in $\tau_i = [\phi_0^1, \phi_0^2, \phi_0^3, \ldots, \phi_0^{N_i-1}]$ to match our GMF-PA model. Frames are ordered such that a suspending frame executes after a computational frame. The minimum inter-arrival time for $\tau_i$ is $P_i$, and the end-to-end deadline is $D_i^{N_i-1} + \sum_{j=0}^{N_i-2} P_i^j$. The end-to-end deadline is different than the one in self-suspending tasks due to the consideration of arbitrary-deadline systems. We also have the cycle deadline $D_i$ that is used to bound the deadlines of frames.

Our parameter-adaption methods that are derived in Section IV for the GMF-PA model are immediately applicable to the FRD-scheduling [9], [13] for multi-segment self-suspending tasks. The details will be explained later.

III. RELATED WORK

In this section, we introduce the related work of the GMF-PA model in Section III-A. As far as we know, there is no related work that assign parameters in the GMF model. Thus, we apply the GMF-PA model to the self-suspending tasks and compare the schedulability ratio and running time with latest work. We survey the related work of self-suspending tasks in Section III-B.

A. The Generalized Multiframe Model

The generalized multiframe model (GMF) was first introduced by Baruah et al. [4] to extend the sporadic task model and multiframe task model (MF) [15]. In the GMF model, the frames are executed in order and thus form a “cycle”. The cycle can recur for infinite times. In the non-cyclic GMF task model [16], the frames can execute out of order and thus reduce the pessimism of the modeling of software-defined radio [20]. The recurring real-time task (RRT) model [5] is a generalization of the GMF model to handle conditional codes. The non-cyclic recurring real-time task model [3] can generalize all the models referred above.

There are many applications based on the MF model and GMF model. Ding et al. [10] scheduled a set of tasks with the I/O blocking property under the MF model. Ekberg et
al. [11] have developed an optimal resource sharing protocol for the GMF model. Andersson [2] has presented a schedulability analysis for the flows in multi-hop networks comprising software-implemented Ethernet switches, according to the GMF model. The GMF model has great advantages and is applied to multiple areas. However, current related models assume that parameters are fixed during the task specification time. Instead in our GMF-PA model, parameters are flexible to be chosen under the frame constraints and cycle constraints. Similar flexible models, such as the parameter-adaption model [8] and elastic model [7], are also used in many applications.

B. The Self-suspending Task Model

We apply the GMF-PA model to self-suspending tasks in this paper. The traditional task system [14] which considers self-suspending delays as parts of computation times sacrifice significant system capacity, especially when the delay of a self-suspension is large. In this case, the scheduling algorithms which explicitly consider the delays of suspensions receive much attention recently, both for the tasks with at most one-segment self-suspending frame and ones with multiple-segment self-suspending frames.

In the system that each task may have one-segment self-suspension frame, Ridouard et al. [19] have proved that scheduling such periodic self-suspending tasks on a uniprocessor is NP-hard in the strong sense. Nelissen et al. [17] also showed that the timing analysis of sporadic self-suspending tasks is also not easy and computed exact worst-case response times using mixed-integer linear programming (MILP). Due to the hardness of such scheduling problems, Chen and Liu [9] have given the fixed-relative-deadline (FRD) scheduling algorithm to improve the schedulability on sporadic self-suspending tasks on uniprocessor, and quantified the quality of their approach by analyzing the speed-up factors with respect to the optimal FRD scheduling and any arbitrary feasible scheduling.

In the system that each task has multiple-segment self-suspending frames, Huang and Chen [13] proposed FRD scheduling on such tasks with fixed priorities. The multiple-segment self-suspending task model explicitly ensures that all frames must work in order. In this paper, we propose the GMF-PA model and parameter-adaption methods that extend FRD scheduling in arbitrary-deadline systems. We also apply the GMF-PA model to multiple-segment self-suspending tasks.

IV. The Exact Deadline/Period Assignment of Generalized Multiframe Tasks in the GMF-PA Model

In this section, we describe the selection of the deadline and period for each frame under our GMF-PA model by using mixed-integer linear programming (MILP) under EDF scheduling. The period and deadline of each frame are flexible to be chosen under the limit of the frame constraints and the cycle constraints. Along with the selection, the algorithm provides a necessary feasibility test for arbitrary real-valued task parameters. We prove the sufficiency and necessity of the test when task parameters are assumed to be integers later.

Mixed-integer linear programming (MILP) is a mathematical optimization model that contains three parts: an objective function, constraint functions, and ranges of variables. A subset of variables can be restricted to integers in MILP. An MILP aims to find the optimal value of the objective function under the restriction of constraint functions. We build our MILP to select the relative deadlines and periods for all frames. At the same time, MILP gives a necessary feasibility test. However, note that for non-integer parameters, since the MILP is only necessary, the returned selection of deadlines and periods may not be feasible. Later, in Section V, we will give an approximation algorithm for the MILP that returns a feasible selection of periods and deadlines for the non-integer case.

We will first introduce the demand bound function and supply bound function which are used for schedulability analysis. The demand bound function $dbf_i(t)$ accounts for the task $τ_i$’s accumulated execution time of jobs which have both release time and deadline inside any interval of length $t$, and the supply bound function $sbf(t)$ gives the lower bound of resources that the system can supply over any interval of length $t$. In general, the sufficient and necessary condition for a single-processor feasible system is Equation 1.

$$\sum_{τ_i ∈ T} dbf_i(t) ≤ sbf(t), \quad \forall t. \tag{1}$$

In a system where parameters are fixed after task specification time, parameters are given as constants. However, in this paper, we let deadlines and periods be variables and will select such parameters using MILP. The demand in this case is also treated as a variable. For instance, if we are trying to determine the demand over some interval $[0, t]$ from some frame $φ_i^k$ that arrives at time zero; if the relative deadline $D_i^k$ is set to be smaller or equal to $t$, then the demand from this job should be $E_i^k$; otherwise the demand should be zero. Figure 3 illustrates a graph of this concept for some fixed time point $t$. The details of notations will be introduced later. Using the above concepts, Equation 1 will become a set of constraint functions that a feasible system must obey to find a relative-deadline/period assignment. In our parameter-adaption algorithm, the supply bound function is $sbf(t) = t$ and the length $t$ of any interval length is an integer. Our MILP can return an assignment if the system is schedulable. That is, we can determine the necessary feasibility of the system and select potential parameters at the same time.

The general steps of our algorithm are as follows. For a given sequence of frames and a time interval of length $t$, we calculate the demand contribution of each frame to that interval length. Adding the demands of all frames generates the demand of a task, and adding the demands of all tasks (over all possible sequences of frames) generates the total demand at the time interval length. The system is schedulable at a time interval length if the demand is smaller than the supply. We check all interval lengths, which are integers, in the algorithm.
For a given interval length \( t \), we need to calculate the demand for every possible sequence of frames of task \( \tau_i \) over any interval of length \( t \). Assume that the first frame of \( \tau_i \) to arrive in such an interval is \( \phi_i^j \) (i.e., the \( j \)'th frame). The demand of any sequence starting with the \( j \)'th frame over a \( t \)-length interval is maximized if the \( j \)'th frame arrives exactly at the start of the interval and subsequent frames arrive as soon as possible (e.g., see Barua et al. [4] for GMF schedulability). To calculate the demand from the \( k \)'th frame in such an interval for the specified sequence, \( y_{i,t}^{j,k} \) represent the demand of this frame. We will calculate \( y_{i,t}^{j,k} \) for all possible \( i, j, k \), and \( t \). For simplicity, we use the "\( \varphi \)" to represent the ranges of variables. The task index \( i \) ranges from 0 to \( n \) - 1. The superscripts \( j \) and \( k \) represent the starting frame and the current frame correspondingly, and both have the ranges from 0 to \( N_i - 1 \). The range of any interval length \( t \) has been shown [4] that the maximum interval length is bounded by 
\[
O(\log n \cdot \frac{U_{\text{cap}}}{U_{\text{cap}} - \max_{\tau \in T}(P_i - D_i^0)})
\]
in Lemma 1. The maximum integer interval length since we do not know deadlines beforehand in our GMF-PA model. We use such abbreviations across this paper. The demand of the task \( \tau_i \) started from \( j \)'th frame in time interval length \( t \) is \( y_{i,t}^{j} \). The maximum demand of \( \tau_i \) among all starting frames is \( y_{i,t}^{j} \).

In the Parameter Selection and Exact Feasibility Test MILP, the notations in bold font are constants and the other notations are variables. Lines 3 to 6 are the basic constraints introduced among all variables. Lines 8 to 10 are the basic constraints. Lines 8 and 9. Line 8 is the constraint function that decides the value of \( \phi_{i}^{j,k} \). The length \( t_b \) in Line 9 is the summation of the previous periods \( \lfloor \frac{\phi_{i}^{j,k}}{P_i} \rfloor \) \( P_i \) and the distance from the starting \( j \)'th frame to \( k \)'th frame 
\[
\sum_{p=0}^{P_i(j+p)} \text{mod} N_i + D_i^k
\]
For example, the length \( t_b = P_i^1 + P_i^2 + D_i^2 + \lfloor \frac{\phi_{i}^{j,k}}{P_i} \rfloor \) \( P_i \) if we consider the interval starting with an arrival of the first frame and ending at the deadline of the third frame. In the inequality of Line 8, the lengths \( t_b \) and \( t \) decide whether the demand of \( k \)'th frame in \( t - \lfloor \frac{\phi_{i}^{j,k}}{P_i} \rfloor \) \( P_i \) will be added to \( y_{i,t}^{j,k} \). The constant \( \text{realmin} \) is the smallest representable positive number. When \( t \geq t_b \), the flag \( x_{i,t}^{j,k} \) must be 1 to let MILP feasible and the demand \( y_{i,t}^{j,k} \) \( P_i \) contributes to \( y_{i,t}^{j} \). When \( t < t_b \), the flag \( x_{i,t}^{j,k} \) can be either 0 or 1. However, the demand \( y_{i,t}^{j,k} \) is overestimated when \( x_{i,t}^{j,k} = 1 \). MILP tends to choose 0 for \( x_{i,t}^{j,k} \) because of the smaller demand, and the details are shown in Lemma 1. Figure 3 shows the staircase function between deadline and demand. Note that the inequality in Line 8 is always correct when \( x_{i,t}^{j,k} \) is 1 and \( t \geq t_b \), and when \( x_{i,t}^{j,k} \) is 0 and \( t < t_b \).

Parameter Selection and Exact Feasibility Test

1. minimize: \( \mathcal{L} \)
2. subject to:
3. \( E_i^k \leq D_i^k \leq D_i^k \), \( \forall i, k \).
4. \( E_i^k \leq P_i^k \leq P_i^k \), \( \forall i, k \).
5. \( D_i^{k} \leq P_i^k + |\frac{t}{P_i^k}| \cdot P_i^k \), \( \forall i, k \).
6. \( \sum_{k=0}^{N_i-1} P_i^k \sum_{k=0}^{N_i-1} D_i^k \leq D_i^k \), \( \forall i \).
7. \( y_{i,t}^{j,k} = x_{i,t}^{j,k} \cdot E_i^k + \lfloor \frac{t}{P_i^k} \rfloor \cdot E_i^k \), \( \forall i, j, k, t \).
8. \( t - t_b \leq x_{i,t}^{j,k} - \lfloor \frac{t}{P_i^k} \rfloor \cdot P_i^k \), \( \forall i, j, k, t \).
9. \( t_b = \sum_{p=0}^{P_i^k(j+p)} \text{mod} N_i + D_i^k + \lfloor \frac{t}{P_i^k} \rfloor \cdot P_i^k \).
10. \( y_{i,t}^{j} = \sum_{k=0}^{N_i-1} y_{i,t}^{j,k} \), \( \forall i, j, t \).
11. \( y_{i,t} \geq y_{i,t}^{j} \), \( \forall i, j \).
12. \( \sum_{i=0}^{n-1} y_{i,t} \leq \mathcal{L} \cdot t \), \( \forall t \).
13. and:
14. \( D_i^k, P_i^k, t_b, y_{i,t}^{j}, y_{i,t}^{j,k}, y_{i,t}^{j}, \mathcal{L} \in \mathbb{R}^{*}, x_{i,t}^{j,k} \in \{0, 1\} \).

In Line 10, the demand \( y_{i,t}^{j} \) of task \( \tau_i \) starts from \( j \)'th frame. In Line 11, the demand \( y_{i,t} \) is the maximum demand for \( \tau_i \) over all possible starting frames. At last, the demand of all tasks \( \sum_{i=0}^{n-1} y_{i,t} \) has to be less than the supply bound function for all \( t \) as showed in Equation 1; otherwise, the system is not schedulable. In Line 12, \( \mathcal{L} \) is set to indicate how schedulable or not schedulable the system is. If the system is schedulable, then \( \mathcal{L} \leq 1 \).

In the setting of our MILP, the variables \( D_i^k, P_i^k, t_b, y_{i,t}^{j,k}, y_{i,t}^{j}, y_{i,t} \), and \( \mathcal{L} \) are free variables. The number of all variables is pseudo-polynomial bounded. The flag \( x_{i,t}^{j,k} \) is restricted to be an integer variable that is either 0 or 1. The relationship among the variables is summarized.
in Figure 4. The boxes with solid lines contain free variables and the boxes with dotted lines contain constants. The arrows show the dependable relationships and the integers on the arrows indicate the number of lines in MILP. For example, Line 6 to 9 show that the constant \( P_i \) has an effect on the variables \( P_i^k \), \( t_b \), \( x_i^{j,k} \) and \( y_i^{j,k} \). All variables are connected and restrained in MILP. Eventually, minimizing \( \mathcal{L} \) also minimizes the total demand \( \sum_{i=0}^{n-1} y_{i,t} \).

![Figure 4. Relations of the parameters.](image)

Next, we will first prove that \( y_i^{j,k} \) in the MILP setting is an exact demand of \( k \)’th frame during the interval length \( t \) started from \( j \)’th frame in Lemma 1. Using Lemma 1, we can show that our demand bound \( y_{i,t} \) is same as the one in the GMF model [4]. Last, we prove that our MILP is also a sufficient and necessary schedulability test for integer parameters in Theorem 2.

**Lemma 1.** The value of \( y_i^{j,k} \) in the MILP is the exact worst-case demand of frames \( \phi_i^k \) over an interval of length \( t \) when the first frame of \( \tau_i \) to arrive in the interval is \( \phi_i^j \) (with respect to the deadline and periods assigned to each frame of \( \tau_i \) by the MILP).

**Proof.** We first prove that \( y_i^{j,k} \) is an upper bound of the demand, then prove \( y_i^{j,k} \) is the exact demand. Worst-case means that the interval length \( t \) starts at the release time of \( j \)’th frame and all succeeding frames release as soon as possible.

For an interval of length \( t \), the demand \( \lfloor \frac{t}{P_i} \rfloor \cdot E^i_k \) contributes to \( y_i^{j,k} \) in the first \( \lfloor \frac{t}{P_i} \rfloor \) cycle periods. The remaining question is that whether the \( k \)’th frame in the \( \lfloor \frac{t}{P_i} \rfloor + 1 \)’th cycle will contribute to the interval length \( t - \lfloor \frac{t}{P_i} \rfloor \cdot P_i \). This is exactly what Line 7 in our MILP tells. The flag \( x_i^{j,k} \) tells whether the \( k \)’th frame in the \( \lfloor \frac{t}{P_i} \rfloor + 1 \)’th cycle will contribute to the interval length \( t - \lfloor \frac{t}{P_i} \rfloor \cdot P_i \). The flag \( x_i^{j,k} \) has the value which is either 0 (not contribute) or 1 (contribute).

Lines 8 and 9 decide the value of \( x_i^{j,k} \). Line 9 shows that \( t_b \) is the interval length that compares with the time interval length \( t \). The time \( \lfloor \frac{t}{P_i} \rfloor \cdot P_i \) tracks \( \lfloor \frac{t}{P_i} \rfloor \) cycles. Since the starting frame of \( t \) is \( \phi_i^j \), the starting stage is still \( \phi_i^j \) after \( \lfloor \frac{t}{P_i} \rfloor \) cycles. Starting from \( \phi_i^j \), the length \( (k-j-1) \mod N_i \), \( P_i^{(j,p)} \mod N_i \) is compared with \( t - \lfloor \frac{t}{P_i} \rfloor \cdot P_i \) to check whether \( \phi_i^k \) in the \( \lfloor \frac{t}{P_i} \rfloor + 1 \)’th cycle contributes. In other words, the inequality \( t \geq t_b \) means that \( \phi_i^k \) in the \( \lfloor \frac{t}{P_i} \rfloor + 1 \)’th cycle will contribute to \( y_i^{j,k} \); Otherwise, the frame does not contribute when \( t < t_b \). Note that we only need to consider the \( \lfloor \frac{t}{P_i} \rfloor + 1 \)’th cycle because \( \lfloor \frac{t}{P_i} \rfloor \cdot P_i \leq t < (\lfloor \frac{t}{P_i} \rfloor + 1) \cdot P_i \). When \( t \geq t_b \), \( x_i^{j,k} \) is forced to be 1 showed in Line 8. When \( t < t_b \), \( x_i^{j,k} \) can be either 0 or 1. Thus, \( y_i^{j,k} \) is an upper bound of the demand when \( x_i^{j,k} \) is 1.

We have proved that \( y_i^{j,k} \) is an upper bound of the demand when \( x_i^{j,k} \) is 1. Now we prove that \( y_i^{j,k} \) is an exact demand. When \( t \geq t_b \), \( x_i^{j,k} \) is forced to be 1 and \( y_i^{j,k} \) is an exact demand. When \( t < t_b \), the system always let \( x_i^{j,k} \) be 0 instead of 1. The reason is that we minimize \( \mathcal{L} \) in the objective function. In Line 12, minimizing \( \mathcal{L} \) minimizes \( y_{i,t} \). Minimizing \( y_{i,t} \) also minimizes \( y_i^{j,k} \) and \( y_i^{j,k} \). In all, \( y_i^{j,k} \) is the exact worst-case demand for \( \phi_i^k \) over the interval length \( t \) stated from \( \phi_i^j \).

**Theorem 1.** The value of \( y_{i,t} \) in the MILP over any interval of length \( t \) is exactly the value of \( \text{cbf}_i(t) \) for the GMF model when the parameters of \( \tau_i \) are assigned the values of the deadline and period variables of the MILP.

**Proof Sketch.** In Baruah et al. [4], the demand of a GMF task \( \tau_i \) over any interval of length \( t \) is calculated by considering worst-case arrival sequence over all possible starting frames. In Lemma 1, we have shown that \( y_i^{j,k} \) is an exact upper bound on demand of \( \phi_i^k \) assuming a starting frame of \( \phi_i^j \) for the assigned parameter of the MILP. In Line 10, the demands of all frames of \( \tau_i \) over this interval (assuming a starting frame of \( \phi_i^j \)) are summed together to determine \( y_{i,t} \). Finally, \( y_{i,t} \) is determined by the maximum value of \( y_i^{j,k} \) over all \( j \); that is, the maximum demand over the \( t \) length interval considering all starting frames. Thus, this is exactly what is computed in Baruah et al. [4].

**Theorem 2.** For arbitrary, real-valued parameters, our MILP is a necessary feasibility test. When the period and deadline parameters must be integers (i.e., \( D_i^k, P_i^k \in \mathbb{N} \forall i, k \)), then the MILP is an exact feasibility test.

**Proof.** The necessity is straightforward to prove. We have shown that the demand for each frame is an exact worst-case demand at any interval length \( t \). If the task system is feasible, \( \sum_{i=0}^{n-1} y_{i,t} \leq \mathcal{L} \cdot t \) and \( \mathcal{L} \leq 1 \).

For integer-constrained values of periods and deadlines, we will prove the sufficient condition by contradiction. Suppose there exists a deadline assignment \( A \) that is a feasible assignment under the GMF-PA model, but \( A \) is not feasible under our MILP. In other words, \( A \) will cause \( \sum_{i=0}^{n-1} y_{i,t} > t \) under MILP. Assume \( A = (A_0^1, A_0^1, \ldots, A_0^{N_1-1}, A_1^1, \ldots, A_1^{N_2-1}, \ldots, A_n^0, \ldots, A_n^{N_n-1}) \), where \( N_i \) is the maximum number of frames for each task and \( n \) is the number of tasks. A pair \( A_i^j = (D_i^j, P_i^j) \) indicates
an assignment of deadline and period for $\varphi_i^j$. The assignment $A$ must satisfy the five basic requirements in the GMF-PA model, and satisfy Line 7, 8, and 9 based on Lemma 1. The demand of assignment $A$ is not larger than the worst-case demand in Line 7. Thus, the assignment $A$ satisfies all the constraints in MILP and $\sum_{i=0}^{n-1} y_{i,t} \leq t$. Here we have made a contradiction from the previous assumption that $\sum_{i=0}^{n-1} y_{i,t} > t$.

Thus, the assignment $A$ is also a feasible assignment under MILP and our MILP method is a sufficient condition when parameters are integers for the feasibility test. In other words, we cannot test all the real-number lengths of $t$. Our algorithm is a necessary condition for schedulability in general and an exact condition when parameters are assumed to be integers. In all, the theorem is proved.

V. THE APPROXIMATION ALGORITHM BASED ON MILP

In the previous section, we have built our MILP which can select the deadlines and periods of GMF-PA tasks under EDF scheduling. The method also indicates a necessary feasibility test at the same time. However, solving an MILP is NP-hard in general. Furthermore, the feasibility of our MILP is coNP-complete that trivially transformed from the feasibility test of sporadic tasks [12]. In this section, we will modify the MILP to obtain an approximation based on reducing the number of time interval lengths being tested$^1$. We also show that the speed-up factor of our approximation algorithm is $1 + \epsilon$ with respect to the exact schedulability test of GMF-PA tasks under EDF.

The number of time instants being tested in MILP is $H$ (defined in Section IV), and the number of time instants being tested in the approximation algorithm is $H_a$. The set of test instants in MILP is $T = \{1, 2, 3, ..., H\}$ and the set in the approximation of MILP is $T_a$. The supply bound function used in MILP is showed in Equation 2.

$$sbf(t) = t. \tag{2}$$

Since the number of variables and equations in MILP depend on $H$, the size of MILP will grows quick when $H$ grows. We propose an approximation method based on reducing the number of time instants. We start from the initial time instant $t_0$. The increasing rate is $\epsilon > 0$. We choose the time instants by the increasing rate; thus, $T_a = \{t_0, t_0 * (1 + \epsilon), t_0 * (1 + \epsilon)^2, ..., t_0 * (1 + \epsilon)^{H_a-2}, H\}$. Note that the $H_a - 2$th element is not larger than $H$, and we add $H$ at the end as the $H_a - 1$th element. Note that the increasing rate between the last two elements is not larger than $\epsilon$. For example, the set $T_a$ is $\{1, 1.5, 2.25, 3.375, ..., 17.0859375, 20\}$ for $H = 20$, $t_0 = 1$ and $\epsilon = 0.5$. The supply $sbf_a(t)$ in the approximation algorithm is showed in Equation 3.

$$sbf_a(t) = \begin{cases} 
0, & 0 \leq t \leq t_0 \\
0 * (1 + \epsilon)^k, & t_0 * (1 + \epsilon)^k < t \leq t_0 * (1 + \epsilon)^{k+1} \\
H, & t = H 
\end{cases} \tag{3}$$

In our $sbf_a(t)$, the starting time instant is $t_0 = \min_{\tau_i \in T} E_{\min}^i$ and the range of integer $k$ is $[1, H_a-2]$ in our approximation algorithm. Figure 5 shows an example of the relationship among $\sum_{\tau_i \in T} dbf_i(t)$, $sbf(t)$ and $sbf_a(t)$. It is straightforward to show that the number of elements in $T_a$ is $O(\log_{1+\epsilon} H)$.

Next, we modify the general schedulability condition of Equation 1 with respect to the reduced set of testing points $T_a$.

![Figure 5](image)

Figure 5. In this Figure, line $y=1$ is the supply bound function ($sbf(t)$) of MILP. The stair case function drawn in dotted line is the supply bound function ($sbf_a(t)$) of an approximation algorithm. The staircase function drawn in dashed line is an example of a demand $dbf(t) = \sum_{\tau_i \in T} dbf_i(t)$. The square points on $sbf_a(t)$ are the only required test instants that is proved in Theorem 3. In this example, the total demand $\sum_{\tau_i \in T} dbf_i(t) \leq sbf(t)$ at all time instants $t$. But, the demand $\sum_{\tau_i \in T} dbf_i(t) > sbf_a(t)$ shown at the red circle.

**Theorem 3.** Consider any task system composed of tasks $T$ (e.g., GMF tasks) where the $dbf_i(t)$ is computable (e.g., see Baruah [4]) for any $\tau_i \in T$. Then, by checking the following modified condition:

$$\sum_{\tau_i \in T} dbf_i(t) \leq sbf_a(t), \quad \forall t \in T_a, \tag{4}$$

where $t_0$ of $sbf_a$ must not be larger than the $min_{i,j} \{D_i^j\}$.

We have the following guarantee:

1) If $\sum_{\tau_i \in T} dbf_i(t) \leq sbf_a(t), \quad \forall t \in T_a$, the system is EDF-schedulable on a unit-speed processor.

2) If $\exists t \in T_a$, $\sum_{\tau_i \in T} dbf_i(t) > sbf_a(t)$, the system is EDF-infeasible on a unit-speed processor.

**Proof.** We first show the sufficiency on unit-speed processors for Equation 4 by contradiction. Assume the task system satisfies Equation 4 but is infeasible. This means that there exists a time interval $t'$ when Equation 1 is violated (i.e., $\sum_{\tau_i \in T} dbf_i(t') > sbf(t')$), since it is a necessary and sufficient condition. Assume that $t' \in$
\((t_0 + (1 + \epsilon)^k, t_0 * (1 + \epsilon)^{k+1})\) for some \(k \in \mathbb{N}\). Note that \(sbf_a(t')\) equals \(sbf(t_0 * (1 + \epsilon)^{k+1})\) which is \(t_0 * (1 + \epsilon)^k\). However, note that it is known that the \(dbf_i\) function is monotonically non-decreasing \([4]\). Thus, if \(\sum_{t_i \in T} dbf_i(t') > sbf(t_0)\), then \(\sum_{t_i \in T} dbf_i(t_0) > (1 + \epsilon)^k sbf(t_0)\) which is a contradiction of Equation 4.

We now prove the infeasibility on a slower processor when Equation 4 is not satisfied. In order to prove the “speed-up factor”, assume \(\sum_{t_i \in T} dbf_i(t') > sbf(t_0)\) at time \(t^*\). It must be that \(t^* > t_0\) since for all values of \(t \leq t_0\) the \(dbf_i(t)\) is zero by supposition that \(t_0\) exceeds the minimum frame relative deadline. Furthermore, it is easy to observe that for all \(t \geq t_0\), the \(sbf(t)\) is at most \((1 + \epsilon)\) times larger than \(sbf(t_0)\). From this, we have:

\[
\max_{t > 0} \frac{\sum_{t_i \in T} dbf_i(t)}{sbf(t)} \geq \frac{\sum_{t_i \in T} dbf_i(t^*)}{sbf(t^*)} \\
\geq \frac{\sum_{t_i \in T} dbf_i(t^*)}{(1 + \epsilon)} \\
\geq \frac{\sum_{t_i \in T} dbf_i(t^*)}{(1 + \epsilon) sbf(t_0)} \\
\geq \frac{1}{1 + \epsilon} \quad \text{(By Assumption)}.
\]

Thus, we have proved that the speed-up factor is \(1 + \epsilon\). □

We can now apply Theorem 3 to modify the MILP to create a sufficient approximate feasibility test for the GMF-PA task model with arbitrary, real-valued parameters. To do so, we simply limit the range of \(t\) to now be \(T_a\) for all constraints that depend upon \(t\), and modify Line 12 of MILP to be \(\sum_{i=0}^{n-1} y_{i,t} \leq L * \frac{t}{1 + \epsilon}\). Clearly, this reduces the number of constraints by a logarithmic factor (dependent upon our choice of \(\epsilon\)). We refer to this approximate assignment algorithm as MILP-\(\epsilon\).

In all, the approximate MILP is a sufficient feasibility test. The number of the time interval lengths being tested is reduced from \(O(H)\) to \(O(\log_{1+\epsilon} H)\). Since the number of variables and number of equations depend on the number of time instants, the running time is greatly reduced. We have done exhaustive experiments in Section VII. Our MILP algorithm and approximation algorithm also work for the multi-segment self-suspending tasks represented by our GMF-PA model. The transformation is presented in the next section.

VI. FIXED-RELATIVE-DEADLINE ASSIGNMENT FOR MULTI-SEGMENT SELF-SUSPENSION TASKS

Before applying the GMF-PA model to multiple-segment self-suspending tasks, we first state the limitation of the deadline assignment in the previous papers \([9\), \([13]\]. Equal-deadline assignment (EDA) assigns each computational frame of \(\tau_i\) the same deadline \(D_i^t = \frac{\tau_i - \sum_{m_i \geq s_i} s_i}{m_i}\). Such assignment is restrictive because the system is already infeasible if \(C_i^t > D_i^t\). The situation may happen in several applications. For example, in computation offloading of a robot car \([18]\), the cost of the first computational frame (image capture) and the suspending frame occupy most of the time in one period, and the cost of the second computation frame (path planning) is relatively small.

In order to generate a fair comparison with the recent results \([9\), \([13]\), our MILP will be transferred to schedule the self-suspending tasks under implicit-deadline system. We set \(D_i^k = P_i^k\) (Line 5 in MILP will be automatically satisfied) and \(D_i = P_i\). The variables \(D_i^k, P_i^k, D_i, P_i\) are reduced to \(D_i^k, P_i\) for all \(i \in T\). In this case, the end-to-end deadline of \(\tau_i\) is \(P_i\). Because the previous work \([9\), \([13]\) have no frame constraints, we set \(P_i^k = P_i^k\) and \(D_i^k = P_i^k\) for any computational frame \(\phi_i^k\). Note that \(D_i^1\) is any suspending frame \(\phi_i^k\) due to the setting introduced in Section II-B. The constraints from Line 3 to 6 thus become:

1) \(P_i^k \leq D_i^k, \forall i, k\).
2) \(\sum_{k=0}^{N_i-1} D_i^k \leq P_i, \forall i\).

Since the evaluations \([9]\) contain the situation when \(U_{cap} = 1\), the range of the interval length \(t\) becomes \([\min \left(\log n * \frac{U_{cap}}{1-U_{cap}}, \max_{t_i \in \mathcal{C}} (P_i - P_i^{min})\right), \text{hyperperiod}\]\).

The task system hyperperiod \(\text{hyperperiod} = \text{lcm}\{P_0, P_1, ..., P_{n-1}\}\) is the least common multiple of the \(n\) tasks’ cycle periods. The other lines in our MILP and approximation algorithms remain same.

VII. EVALUATION

We have implemented our MILP and approximation algorithm MILP-\(\epsilon\) using the commercial solver GUROBI \([1]\) in MATLAB. GUROBI is a state-of-the-art mathematical programming solver that has great performance in solving linear and mixed-integer programming problems.

As far as we know, there is no previous work to directly compare with the flexible generalized multiframe model in this paper. We compare our work with the application to self-suspending tasks on uniprocessor \([9\), \([13]\). We restrict our model to compare with one-segment self-suspending tasks using EDA under EDF scheduling \([9]\). We also compare our work with multiple-segment self-suspending tasks \([13]\) using EDA under FP scheduling. Since EDF scheduling is optimal under uniprocessor systems, we compare EDA and our algorithms for multiple-segment self-suspending tasks under EDF. MILP-0.0 is our necessary feasibility test and MILP-\(\epsilon\) (\(\epsilon > 0\)) is our approximation algorithm. The algorithms used in the previous paper \([9\), \([13]\) are EDA and EDA-linear \([9]\). The algorithm EDA assigns each task the same deadline for each computational frame and EDA-linear is EDA’s approximation algorithm.

For one-segment self-suspending tasks, we follow the similar setting of the previous paper \([9]\). Tasks are randomly generated. There are three ranges for task utilizations and the length of self-suspensions. The ranges for task utilization \(U_i\) are \([0.01, 0.1]\) (I, short for “light”), \([0.1, 0.3]\) (m short
for “medium”), and [0.3, 0.6] (l, short for “long”).

The ranges for the length of suspensions are [0.01 * (1 – U_i) * T_i, 0.1 * (1 – U_i) * T_i] (s, short for “short”), [0.1 * (1 – U_i) * T_i, 0.3 * (1 – U_i) * T_i] (m, short for “medium”), and [0.3 * (1 – U_i) * T_i, 0.6 * (1 – U_i) * T_i] (l, short for “long”).

Tasks are randomly generated until the total utilization is equal to \( U_{cap} \). The utilization of the last task is reduced if the total utilization is larger than the cap.

Due to the space constraint, we show the results for medium utilization and medium suspension length in two different settings. The results of the other combinations follow a similar pattern. For the reason that the least common multiple of periods could be very large, e.g., \( 2 \times 10^8 \) for the period range [1, 20]. We first generate Figure 6(a) and 6(b) in which the tasks have small cycle periods that randomly chosen in [1, 10]. Such task systems are randomly generated 50 times under each \( U_{cap} \) and the running time in the figure is the average running time. The schedulability ratio is the number of feasible systems over the total systems generated. MILP and MILP-0.1 have higher schedulability ratio than EDA and EDA-linear, but have longer running times as shown in Figure 6(b). MILP can schedule at most 38 (64) percent more than EDA (EDA-linear).
linear) when $U_{\text{cap}} = 1 \ (0.9)$. MILP-0.1 can schedule at most 16 (44) percent more than EDA (EDA-linear) when $U_{\text{cap}} = 1 \ (0.9)$. MILP (MILP-0.1) uses at most around 184(70) seconds more than EDA and EDA-linear when $U_{\text{cap}} = 1.$ In order to generate a set of tasks with larger hyperperiods, we have the experiments shown in Figure 6(c) and 6(d). The cycle period of each task is randomly chosen in $[1,50]$ and has lower and upper bounds. Such task system is randomly generated 20 times at each utilization cap. In the figure, MILP-0.1 behaves better on both schedulability ratio and running time.

For multiple-segment self-suspending tasks, we follow the similar setting of the previous paper [13]. Due to high execution times for both EDA and MILP, we apply the same approximation of Theorem 3 to EDA and compare with our MILP-ε. The cycle periods $\mathcal{P}_i$ are randomly generated from $[0,1000]$. Under each $U_{\text{cap}}$, ten tasks are randomly generated. The UUniFast algorithm [6] is used to divide the utilizations $U_i$ of the ten tasks under $U_{\text{cap}}$. The total execution time $C_i = \mathcal{P}_i \times U_i$, and the total suspension delay is generated under the three ranges which are the same as the ones in one-segment self-suspending tasks [9]. The UUniFast algorithm is also used to divide total execution times and suspension lengths to the ones for multiple frames. Figure 7 shows that our MILP-ε is better than EDA-ε for both two and five suspending frames. Note that the utilization, execution time and suspending length are uniformly distributed. The result of EDA in other distributions may be even worse since the deadlines are equally assigned.

In all, our MILP and MILP-ε algorithms always yield higher schedulability ratio on both one-segment and mult-segment self-suspending tasks. MILP-ε has the lowest running time when the hyperperiod is large.

VIII. CONCLUSION

Upon the GMF model, we propose the GMF-PA model which has frame constraints and cycle constraints to let the deadline and period of each frame be flexible. Using the mixed-integer linear programming (MILP), we propose an algorithm that can select deadlines and periods in the GMF-PA model. Our MILP-based algorithm is an exact feasibility test when parameters are integers, and a necessary feasibility test in general. In order to reduce the running time of the MILP algorithm, we propose an approximation algorithm MILP-ε based on the supply bound function. The number of time instants being tested is bounded by a logarithmic function of the task system parameters. We prove that the MILP-ε is a sufficient feasibility test.

We apply our MILP and MILP-ε to self-suspending tasks. We remove the assumption that the deadlines are fixedly equally assigned in the previous work. Exhaustive experiments for both one-segment and multiple-segment self-suspending tasks have shown that our algorithms have improved the schedulability ratio and running time compared to the previous results.

In the future, we will work to further improve the efficiency of our algorithm by considering other optimization techniques that remove the integer requirement of our MILP, which are able to determine whether the results are unbiased. We may also compare our work with transaction-based models in distributed systems. Our overall goal is an algorithm that can be used as an online optimization technique for determining parameters in an interactive real-time system design framework.

REFERENCES