ILP representation for Limited Preemption in Energy-Neutral Single-Core Systems

Pierre-Emmanuel Hladik, Houssam-Eddine Zahaf, Sébastien Faucou, Audrey Queudet

▶ **To cite this version:**

Pierre-Emmanuel Hladik, Houssam-Eddine Zahaf, Sébastien Faucou, Audrey Queudet. ILP representation for Limited Preemption in Energy-Neutral Single-Core Systems. The 32nd International Conference on Real-Time Networks and Systems, Nov 2024, Porto, Portugal. 10.1145/3696355.3696366 . hal-04729481

HAL Id: hal-04729481
https://hal.science/hal-04729481v1

Submitted on 10 Oct 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ILP representation for Limited Preemption in Energy-Neutral Single-Core Systems

Pierre-Emmanuel Hladik
Sébastien Faucou
Houssam-Eddine Zahaf
Audrey Queudet*

Nantes Université, École Centrale Nantes, CNRS, LS2N,
UMR 6004, F-44000 France
firstname.lastname@ls2n.fr

ABSTRACT

In this work, we present an Integer Linear Programming (ILP) based approach that optimally selects preemption points for a set of real-time tasks. Beyond to meeting real-time constraints, the system must also consider energy availability constraints to ensure energy neutrality. This enables the design of autonomous low-end IoT devices with real-time constraints, minimizing maintenance operations to battery replacements.

We demonstrate that our system, despite being modeled using ILP, achieves strong temporal performance, even for large-scale problems. We evaluated its performance using the Gurobi solver across a wide range of synthetic experiments.

ACM Reference Format:
Pierre-Emmanuel Hladik, Houssam-Eddine Zahaf, Sébastien Faucou, and Audrey Queudet. 2024. ILP representation for Limited Preemption in Energy-Neutral Single-Core Systems. In *International Conference on Real-Time Networks and Systems (RTNS '24), November 7–8, 2024, Porto 0, Portugal.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3696355.3696366

1 INTRODUCTION

One of the key challenges in designing low-end IoT devices is maximizing device lifespan.A significant limitation impacting system longevity is energy availability. When the system operates without a wired power supply, efficient energy management becomes crucial. Two main approaches are commonly used. In the first, the system designer maximizes the system's lifetime by reducing energy consumption through minimizing operating frequencies and controlling sleep modes. In this approach, the battery capacity must be calibrated to meet the system's lifetime objectives. Once the batteries are exhausted, maintenance operations to replace them are required. This approach is feasible when the device is accessible. However, for inaccessible low-end devices, battery replacement can be costly or even impossible. The second approach focuses on designing an energy-neutral system with a significantly extended lifespan. Such a system must harvest energy from its surrounding environment and rely only on this available energy to operate. In scenarios with limited energy availability, it is crucial to optimize energy consumption by carefully scheduling workloads based on energy availability.

Based on energy availability and predictability, two categories of energy-neutral systems can be designed:

- Systems Allowing Failures: These systems are typically designed for environments where energy availability is *random*. In the literature, numerous works have addressed this by inserting checkpoints to save the task execution state in permanent memory (e.g., FRAM). This approach prevents replaying previously executed segments of the task code. If an energy failure (runout of energy) occurs during the execution of a segment, that segment must be replayed.
- Systems Avoiding Failures: the systems are designed to prevent failures during workload execution, eliminating the need to replay any part of the task. This requires accurate knowledge of energy availability to ensure the system consistently maintains sufficient energy to complete each execution without interruption.

When these devices operate under real-time constraints, it is essential to have an accurate understanding of the energy availability profile. Either the complete energy harvesting profile is known in advance, allowing for clairvoyant strategies to be built, or minimal guarantees about the harvested energy are ensured. Optimal preemptive schedulers, such as ED-H [15], have been proposed in the literature. In such systems, a task is guaranteed to be executed when both schedulability and energy constraints are satisfied. This approach assumes tasks can resume execution as soon as energy becomes available, without factoring in the preemption costs linked to resuming execution or the constraints of saving task states. Consequently, these approaches remain theoretical, and implementing them would require significant modifications, resulting in the loss of optimality.

A practical solution involves enabling non-preemptive scheduling. In this scenario, once a task begins execution, it runs to completion. However, fully executing a task may require more energy than the storage device can provide. One solution is to decompose the task execution into multiple basic blocks. Each basic block must execute to completion. By inserting charging times between blocks when required and/or desired, it is possible to ensure that all basic blocks will always execute to completion. The scheduler can be invoked at the completion of each basic block, allowing

*Both authors contributed equally to this research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
RTNS '24, November 7–8, 2024, Porto 0, Portugal
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1724-6/24/11…$15.00
https://doi.org/10.1145/3696355.3696366

for preemption. Therefore, each basic block can be associated with a preemption cost, including permanent context saving, resuming, cache-related preemption delays, etc. Generally, more basic blocks increase system robustness. However, this also increases the total cost of preemption and may increase the complexity of system analysis.

Contributions. In this work, we select preemption points to avoid preempting a task at every basic block, reducing both temporal and energy preemption costs. We ensure that, under energy availability constraints, all defined non-preemptive regions—a sequence of basic blocks—will always execute to completion while respecting timing constraints.

We model the problem of selecting preemption points while respecting real-time and energy availability constraints using an Integer Linear Program (ILP). To avoid the explosion of real-time constraints, we use a parameterized scheduling constraint inspired by [8]. This approach over-constrains the system *without loss of optimality*, handling the intrinsic complexity of the problem.

We demonstrate that our system, although modeled using an ILP, is capable of achieving good temporal performance even for large-scale problems. We use the Gurobi solver on a large set of synthetic experiments.

Paper structure. The article begins by presenting the model employed to represent the problem, then introduces in Section 4.3 a scheduling analysis method for the studied problem in the form of a parameterized version. The subsequent section provides a detailed account of the formulation of energy constraints. Section 5 provides a brief overview of existing works. Section 6 focuses on presenting the conducted experiments and the obtained results, before concluding in the final Section 7.

2 SYSTEM MODEL

2.1 Overview of the system

Designing an energy harvesting system involves the integration of several key hardware and software components to efficiently capture, convert, store, and manage ambient energy from various sources. A typical design of our target systems is illustrated in Figure 1. The **Transducer** converts ambient energy from the environment (input) into usable electrical energy. These transducers can be piezoelectric (vibrations), RF (Radio Frequency), electromagnetic, or other types. Depending on the energy sources, transducers are able to harvest varying amounts of energy, which might be subject to fluctuations. On the one hand, solar transducers are efficient in harvesting energy from solar radiation, but they are usually subject to significant fluctuations. On the other hand, when harvesting energy from RF sources, this energy is typically available with reduced fluctuation within cities. Similar characteristics apply to transducers harvesting energy from vibrations, hydrokinetics, etc. We use the latter types of transducers and energy sources. The **Rectifier** is designed to maximize energy extraction from the transducer. The **Regulator** is designed to maintain a stable voltage for the system. "When the energy source is predictable, such as those cited above, and by using voltage regulators, it is possible to guarantee a lower bound of power. The energy is either used to replenish a battery/capacitor or directly supply the single-core board, which is itself connected to different I/O (sensors, wireless communication output, etc.).

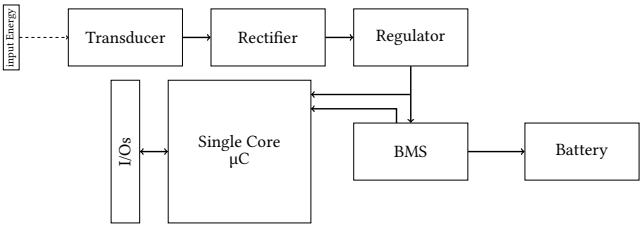

Figure 1: Typical EH system design

The charging profile of a battery depends on the design of the **Battery Management System (BMS)**. A typical design for the BMS of low-end devices such as those operated by lithium-ion batteries, uses a CC-CV (Constant Current-Constant Voltage) charging profile. In CC-CV, BMS provides a constant current and monitors the evolution of the battery voltage. Under constant current, the increase in voltage is almost linear [22].

From a software level, the system is managed using an operating system that executes several tasks, within their respective deadline. Each task is made up of a set of basic blocks that are executed sequentially. A basic block executes atomically, without possible preemption. In this work, we assume that conditionals and loops are confined within the basic blocks.

The beginning of a basic block represents a potential preemption point. A preemption point will be considered active if it allows task preemption (to generalize the approach, we consider that the initial basic block of a task is active). If a preemption point is not active, then the two basic blocks surrounding that point are executed sequentially without possible preemption.

In addition, the energy consumption of tasks is taken into account. Each basic block is thus characterized by its worst-case consumption, and the system is equipped with an energy storage device, which can be a battery or a capacitor. This device is characterized by a maximal storage capacity. Moreover, the system is able to harvest energy from the environment, and we suppose that we know a lower bound on the amount of power that can be harvested at every time instant.

2.2 Task model

A system is composed of a set of n sporadic independent tasks $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$. Each task τ_i is characterized by a tuple $(\mathcal{G}_i, d_i, p_i)$, with:

- $\mathcal{G}_i = \{\gamma_{i,1}, \ldots, \gamma_{i,n_i}\}$ the list of $n_i = |\mathcal{G}_i|$ basic blocks. The tuple $(c_{i,j}^{\text{worst}}, c_{i,j}^{\text{best}}, c_{i,j}^{\text{over}}, e_{i,j}^{\text{worst}}, e_{i,j}^{\text{over}}, v_{i,j}^{\bullet})$ characterizes a basic block $\gamma_{i,j}$ with :
  - $c_{i,j}^{\text{worst}}$ its worst-case execution time;
  - $c_{i,j}^{\text{best}}$ its best-case execution time;
  - $c_{i,j}^{\text{over}}$ its temporal preemption overhead if $\gamma_{i,j+1}$ starts with an active preemption point;
  - $e_{i,j}^{\text{worst}}$ its worst-case energy consumption;
  - $e_{i,j}^{\text{over}}$ the energy consumption overhead if $\gamma_{i,j+1}$ starts by an active preemption point;

– $v^{\bullet}_{i,j}$ the relative difference between harvested and consumed energy during the basic block execution (see below).

Note that the temporal overhead and energy consumption of preemption are already taken into account in the last basic block so c^{over}_{i,n_i} and e^{over}_{i,n_i} are zero.

- p_i: the period of the task. It represents the minimum amount of time that separates two consecutive activations of τ_i.
- d_i: the relative deadline of the task, *i.e.*, γ_{i,n_i}, the last basic block of τ_i, must complete no later than d_i after the last activation of $\gamma_{i,1}$. We consider constrained deadlines $d_i \leq p_i$. We also assume that tasks are ordered by non-decreasing deadlines, *i.e.*, $d_i \leq d_{i+1}$.

2.3 Energy model

The device is characterized by its maximal energy storage capacity, denoted as $v^{\max}$. The lower bound on the amount of power that can be harvested at every time instant is denoted as w^{load}, and the level of energy in the storage device at time instant t is denoted as $\mathrm{v}(t)$. Corresponding to the earlier discussion (see Section 2.1), we assume linear charging profiles. This means that the quantity of energy available to execute tasks increases linearly with time.

Therefore, the energy level in the energy storage device at time instant $t + \Delta t$ (with no consumption between t and $t + \Delta t$) is computed as follows:

$$\mathrm{v}(t + \Delta t) = \mathrm{v}(t) + \Delta t \cdot w^{\text{load}} \tag{1}$$

For a basic block $\gamma_{i,j}$ we define its relative difference in energy consumed, $v^{\bullet}_{i,j}$, which represents the minimum energy the system could harvest during the execution of $\gamma_{i,j}$ minus the maximum energy consumed by the basic block:

$$v^{\bullet}_{i,j} = w^{\text{load}} \cdot c^{\text{best}}_{i,j} - e^{\text{worst}}_{i,j} \tag{2}$$

The term on the left side of the equality represents the minimum energy that will be harvested during the execution of the basic block, and the term on the right side represents its maximum consumption.

A basic block is said to be *energetically positive* if $v^{\bullet}_{i,j} \geq 0$, *i.e.*, its energy consumption is always smaller than the energy that can be harvested by the system during basic block's execution, and *energetically negative* otherwise.

3 OUR SCHEDULER FOR ENERGY HARVESTING SYSTEMS

We consider the Earliest Deadline First (EDF) algorithm to schedule active tasks. The scheduler is activated at each active preemption point and determines the highest priority task based on absolute deadlines. It then checks if the energy storage has enough power to execute this task until the next active preemption point (or its termination). If sufficient energy is available, the task runs immediately until the next active preemption point. If not, a charging period begins until enough energy is accumulated. The scheduler does not re-invoke at the end of charging periods, which are treated as part of task execution. We provide methods to calculate the required energy at each preemption point.

The processor operates in three states: (i) busy, where it executes task code while simultaneously harvests and discharges energy; (ii) charging, where it is assigned a task but not executing, allowing the energy storage to recharge; and (iii) idle, where it is not executing any task but is still charging the storage device. Throughout the system's operation, the energy storage's accumulated energy never exceeds its maximum capacity.

Example 3.1. Let us consider a task system composed of two tasks. Task τ_1 has a period of 20 and a deadline of 8, while Task τ_2 has a period of 30 with a deadline of 15 (see Table 1).

We assume that all preemption points are active, resulting in 2 active preemption points for τ_1 and 3 for τ_2. We illustrate a schedule scenario in Figure 2 considering only one instance per task. The worst-case energy consumed to execute the different blocks is reported in Table 1. We consider that the energy and execution time of preemption points are zero.

In this example, we consider w^{load} equal to 0.5 and $v^{\max}$ equals 2. Gray dashed boxes denote the charging periods, and the green solid ones denote the non-preemptive blocks execution.

Table 1: Temporal and energy characteristics of exampe tasks

Task	Basic blocks	$c^{\text{worst}}_{i,j}$	$c^{\text{best}}_{i,j}$	$e^{\text{worst}}_{i,j}$	$v^{\bullet}_{i,j}$
τ_1	$\gamma_{1,1}$	3	1	2.5	-2
	$\gamma_{1,2}$	2	1	2.5	-2
τ_2	$\gamma_{2,1}$	1.5	1	2	-1.5
	$\gamma_{2,3}$	2.5	2	3	-2
	$\gamma_{2,3}$	3	2	1	0

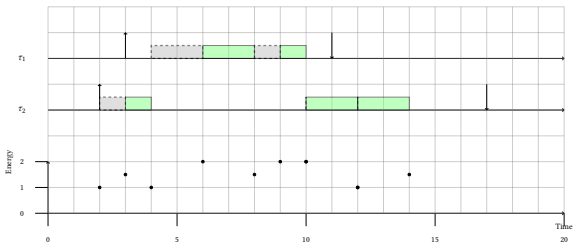

Figure 2: Example an energy-constrained EDF scheduling

The instance of τ_2 is activated first at time $t = 2$. At this time, the energy level in the storage is 1. The first non-preemptive block $\gamma_{2,1}$ requires a storage level of 1.5 to ensure it finishes its execution. Therefore, a charging period of 1 time unit is inserted before the task starts its execution. At the completion of the non-preemptive block $\gamma_{2,1}$, the energy level in the storage device is 1 (at time $t = 4$). Note that the block's consumption was lower than expected, so the storage level is higher than anticipated.

The high-priority task τ_1 arrives at time $t = 3$ when the non-preemptive block of the low-priority task is executing. Since the scheduler has already scheduled a non-preemptive block of τ_2, it does not preempt it. Preemption occurs only at the completion of $\gamma_{2,1}$. The required energy for $\gamma_{1,1}$ is 2, so another charging period is inserted. Once the high-priority task completes its execution, the low-priority task can resume its execution.

The required energy for the second non-preemptive block of the low-priority task is equal to 2, which is already available in the storage device. Therefore, non-preemptive block $\gamma_{2,2}$ starts its execution immediately. When it completes, the energy storage level is equal to 1. The last non-preemptive block of τ_2 does not require a charging period, as it allows the harvesting of more energy than it consumes.

4 ILP FORMULATION FOR PREEMPTION POINTS SELECTION

In this section, we present our ILP formulation. We outline the decision variables, describe the various constraints and methods used to linearize them, and explain how schedulability and energy harvesting constraints are incorporated into our ILP model.

4.1 Preemption point activation

First, we introduce binary decision variable $\mathbf{a}_{i,j}$. It determines whether or not a preemption point is active. The jth preemption point of task τ_i is active if and only if $\mathbf{a}_{i,j}$ equals 1.

$$\mathbf{a}_{i,j} \in \{0,1\} \qquad : i \in [1..n], j \in [1..n_i+1]$$

Please note that the first preemption point is always selected, that is $\mathbf{a}_{i,1} = 1$. Additionally, we define a new preemption point at the end of the last basic block to indicate task completion. This preemption point is also always selected, therefore $\mathbf{a}_{i,n_i+1} = 1$.

4.2 Region

We denote by $\Upsilon_{i,j}$ the set of successive basic blocks that start with $\gamma_{i,j}$ and end at the first activated preemption point[1]:

$$\Upsilon_{i,j} = \{\gamma_{i,k} | j \leq k \leq \min\{l | j \leq l \leq n_i \wedge \mathbf{a}_{i,l+1} = 1\}\} \tag{3}$$

Please note that in this definition, regions do not necessarily begin with an active preemption point. Therefore, it is possible that all basic blocks of region $\Upsilon_{i,j}$ are included in larger region $\Upsilon_{i,k}$, such that $k < j$. This formulation allows us to easily express the different constraints.

We introduce an integer variable $\mathbf{c}_{i,j}$ to denote the sum of the worst-case execution times of basic blocks of the region $\Upsilon_{i,j}$ and the execution overhead of its last basic block.

The variable $\mathbf{c}_{i,j}$ is defined recursively, starting from the last region, where $\mathbf{c}_{i,n_i}$ is by definition equal to c^{worst}_{i,n_i}. The other values of $\mathbf{c}_{i,j}$ are then calculated based on the value of $\mathbf{c}_{i,j+1}$ and depending on $\mathbf{a}_{i,j+1}$. If the preemption point is not active (*i.e.* $\mathbf{a}_{i,j+1} = 0$), then the value of $\mathbf{c}_{i,j+1}$ is simply added to $c^{\text{worst}}_{i,j}$ without considering the overhead induced by the preemption point. On the contrary, if the activation point is active, the execution duration of the region is simply equal to its worst-case execution time plus the overhead induced by the preemption point:

$$\mathbf{c}_{i,n_i} = c^{\text{worst}}_{i,n_i} \tag{4}$$

$$\mathbf{c}_{i,j} = \begin{cases} c^{\text{worst}}_{i,j} + c^{\text{over}}_{i,j} & \text{if } \mathbf{a}_{i,j+1} = 1 \\ c^{\text{worst}}_{i,j} + \mathbf{c}_{i,j+1} & \text{otherwise} \end{cases} \qquad : j \in [1..n_i - 1] \tag{5}$$

[1]In the worst-case, it encounters the last basic block of task τ_i, as it is always activated

For each region $\Upsilon_{i,j}$, we also introduce a variable $\mathbf{b}_{i,j}$ representing the time required to reach a sufficient energy level to execute the region without starvation, assuming an initial energy level of zero. For instance, we assume that this value is computed. We will show further the techniques to compute the worst-case (largest) charging time for a given region.

We define the variable $\mathbf{z}_{i,j}$ as the total duration for the execution of region $\Upsilon_{i,j}$. It is computed as the sum of the required charging time to reach a sufficient energy level and its worst-case execution time:

$$\mathbf{z}_{i,j} = \begin{cases} \mathbf{c}_{i,j} + \mathbf{b}_{i,j} & \text{if } \mathbf{a}_{i,j} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

We highlight that Equation (6) considers only regions that start with an active preemption point.

We introduce the variable $\mathbf{q}_i$ to denote the largest total duration for all regions for task τ_i:

$$\mathbf{q}_i = \max_{j=1}^{n_i} \{\mathbf{z}_{i,j}\} \tag{7}$$

This variable will be used later to assess the system's schedulability, as it represents the maximum blocking time that a task might impose on higher-priority tasks.

Finally, we denote by $\mathbf{w}_i$ the worst-case busy time of a task τ_i, defined as the sum of the duration of all regions:

$$\mathbf{w}_i = \sum_{j=1}^{n_i} \mathbf{z}_{i,j} \tag{8}$$

4.3 Schedulability analysis constraints

In this section, we will show how the schedulability constraints are expressed using linear constraints. To formulate the schedulability test, we draw inspiration from the work of Bertogna *et al.* [9], adapting it to our problem to take into consideration the energy aspects.

We start by introducing the demand bound function for a task τ_i at instant t as:

$$\text{dbf}_\text{i}(\text{t}) = \left(1 + \left\lfloor \frac{\text{t} - \text{d}_\text{i}}{\text{p}_\text{i}} \right\rfloor\right) \mathbf{w}_\text{i} \tag{9}$$

and its maximum blocking time:

$$\mathbf{B}_i = \begin{cases} 0 & \text{if } i = n \\ \max_{i<k\leq n}\{\mathbf{q}_k\} & \text{otherwise} \end{cases} \tag{10}$$

Note that the task set is sorted in increasing order of tasks' deadlines, and so $\mathbf{B}_i$ denotes the maximum blocking time caused by tasks with larger deadlines, i.e., lower priority.

We define the set $\mathcal{D}$ as:

$$\mathcal{D} = \{k \cdot p_j + d_j | k \in \mathbb{N}, j \in [1..n]\} \tag{11}$$

and the value d_{n+1} as:

$$d_{n+1} = \min\left[H, \max\left(d_n, \frac{1}{1 - \sum_{i=1}^{n} \frac{\mathbf{w}_i}{p_i}} \cdot \sum_{i=1}^{n} \frac{\mathbf{w}_i}{p_i}(p_i - d_i)\right)\right] \tag{12}$$

where H is the hyperperiod, *i.e.* the least common multiple of p_1, p_2, …, p_n. The value d_{n+1} represents an upper bound on the time

window within which it is necessary to check the schedulability of a task.

Theorem 4.1. *Task set $\mathcal{T}$ is schedulable with limited preemption EDF if for all $i \in [1..n]$ and for all $t \in \mathcal{D}$ such that $d_i \leq t \leq d_{i+1}$*

$$\mathbf{B}_i \leq t - \sum_{j=1}^{n} \mathrm{dbf_j(t)} \tag{13}$$

Proof. According to the condition in the theorem, the processor time demand, including charging periods, is less than the available processor time. If the sum of execution time and charging periods are schedulable, and since EDF is sustainable, then the timing constraints are respected.

By definition, including charging time in the total task duration prevents every non-preemptive region from energy starvation, thus respecting the energy constraints.

Therefore, our task set is schedulable as both energy starvation and real-time constraints are respected.

□

In the design of our ILP, it is mandatory to express the schedulability and energy constraints using linear constraints. In our case, d_{n+1} is not linear because the variables $\mathbf{w}_i$ are present in both the numerator and denominator of the max. Additionally, since d_{n+1} can potentially be very large, we have considered other approaches.

Several methods to express EDF schedulability tests using ILP have been proposed by Baruah et al. in [8] such as a parametric schedulability test. It computes the demand bound function using either the exact value or an approximation, depending on a certain value of k that expresses the level of approximation. This is expressed as follows:

$$\mathrm{dbf_j^{(k)}(t)} = \begin{cases} \left(1 + \left\lfloor \frac{t-d_j}{p_j} \right\rfloor\right) \mathbf{w}_j & \text{if } t \leq (k-1)p_j + d_j \\ \left(1 + \frac{t-d_j}{p_j}\right) \mathbf{w}_j & \text{otherwise} \end{cases}$$

In the following theorem, we demonstrate that incorporating energy starvation constraints does not invalidate the dbf approximation.

Theorem 4.2. *A restricted-preemption sporadic task system $\mathcal{T} = \{\tau_1, ..., \tau_n\}$ is schedulable under limited preemption EDF if:*

$$\forall i \in [1..n], \forall t \in S_k, \mathbf{B}_i \leq t - \sum_{j=1}^{n} \mathrm{dbf_j^{(k)}(t)} \tag{14}$$

where

$$\mathcal{S}_k = \bigcup_{j=0}^{n} \bigcup_{h=0}^{k} \{d_j + hp_j\}$$

Proof. Assume the set $\mathcal{D} = \{D_1, \ldots, D_{|\mathcal{D}|}\}$ is ordered, *i.e.*, $D_i \leq D_{i+1}$. By definition, we have $\mathcal{S}_k \subset \mathcal{D}$ and $D_1 \in \mathcal{S}_k$.

Assume that for $D_l, 1 \leq l < |\mathcal{D}|$ we have $\forall i \in [1..n], \mathbf{B}_i \leq D_l - \sum_{j=1}^{n} \mathrm{dbf_j^{(k)}(D_l)}$ (this holds for D_1) and that Equation (14) is true.

There are two cases for D_{l+1}:

- If $D_{l+1} \in \mathcal{S}_k$: $\forall i \in [1..n], \mathbf{B}_i \leq D_l - \sum_{j=1}^{n} \mathrm{dbf_j^{(k)}(D_{l+1})}$.
- If $D_{l+1} \notin \mathcal{S}_k$: There exists $(\iota, \kappa) \in [1..n] \times \mathbb{N} such that D_{l+1} = \kappa p_\iota + d_\iota$ and $D_{l+1} > kp_\iota + d_\iota$ (otherwise D_{l+1} would be in $\mathcal{S}_k$). By definition, we have $\mathrm{dbf_\iota^{(k)}(D_l)} = \left(1 + \frac{\mathrm{D_l - d_\iota}}{\mathrm{p_\iota}}\right) \mathbf{w}_\iota$ and $\mathrm{dbf_\iota^{(k)}(D_{l+1})} = \left(1 + \frac{\mathrm{D_{l+1} - d_\iota}}{\mathrm{p_\iota}}\right) \mathbf{w}_\iota$.
  Furthermore, there is no tuple (ι', κ') in $[1..n] \times \mathbb{N}$ with $\iota' \neq \iota$ such that $D_l \leq \kappa' p_{\iota'} + d_{\iota'} < D_{l+1}$. Hence, for $\iota' \neq \iota$:
  - if $\mathrm{dbf_{\iota'}^{(k)}(D_l)} = \left(1 + \frac{\mathrm{D_l - d_{\iota'}}}{\mathrm{p_{\iota'}}}\right) \mathbf{w}_{\iota'}$ then $\mathrm{dbf_{\iota'}^{(k)}(D_{l+1})} = \left(1 + \frac{\mathrm{D_{l+1} - d_{\iota'}}}{\mathrm{p_{\iota'}}}\right) \mathbf{w}_{\iota'}$
  - if $\mathrm{dbf_{\iota'}^{(k)}(D_l)} = \left(1 + \left\lfloor \frac{\mathrm{t - d_{\iota'}}}{\mathrm{p_{\iota'}}} \right\rfloor\right) \mathbf{w}_{\iota'}$ then $\mathrm{dbf_{\iota'}^{(k)}(D_{l+1})} = \mathrm{dbf_{\iota'}^{(k)}(D_l)}$.

From these results, we have:

$$\sum_{j=1}^{n} \mathrm{dbf_j^{(k)}(D_{l+1})} \leq \sum_{j=1}^{n} \mathrm{dbf_j^{(k)}(D_l)} + (\mathrm{D_{l+1} - D_l}) \sum_{j=1}^{n} \frac{\mathbf{w_j}}{\mathrm{p_j}}$$

To be schedulable, it is necessary that $\sum_{j=1}^{n} \frac{\mathbf{w}_j}{p_j} \leq 1$, thus:

$$\sum_{j=1}^{n} \mathrm{dbf_j^{(k)}(D_{l+1})} \leq \sum_{j=1}^{n} \mathrm{dbf_j^{(k)}(D_l)} + (\mathrm{D_{l+1} - D_l})$$

and:

$$D_{l+1} - \sum_{j=1}^{n} \mathrm{dbf_j^{(k)}(D_{l+1})} \geq \mathrm{D_l} - \sum_{j=1}^{n} \mathrm{dbf_j^{(k)}(D_l)} \geq \mathbf{B_i}$$

This leads to the conclusion that Equation (14) implies Equation (13). □

4.4 Energy constraint

The problem is to identify a set of active preemption points that guarantee the system is schedulable. In addition to timing constraints, it is also necessary to evaluate the duration $\mathbf{b}_{i,j}$, which represents the minimum charging time to guarantee that the region $\Upsilon_{i,j}$ executes without starvation.

A region starting with an active preemption point is said to be in energy starvation if it is not possible to guarantee a minimal energy threshold that allows completing its execution without interruption. In other words, a region is not in starvation if there exists an energy storage threshold from which it is possible to end-to-end execute all the basic blocks of the region.

We begin by introducing the variable $\mathbf{v}_{i,j}$, which represents the energy level at the end of the execution of a basic block, assuming that the energy level of the energy storage was zero at the previous active preemption point.

The energy level $\mathbf{v}_{i,j}$ after execution of the basic block $\gamma_{i,j}$ is defined as:

$$\mathbf{v}_{i,1} = \min\{v^{\max}, v^{\bullet}_{i,j} - \mathbf{a}_{i,2} \cdot e^{\mathrm{over}}_{i,1}\} \tag{15}$$

$$\mathbf{v}_{i,j} = \begin{cases} \min\{v^{\max}, v^{\bullet}_{i,j} - \mathbf{a}_{i,j+1} \cdot e^{\mathrm{over}}_{i,j}\} & \text{if } \mathbf{a}_{i,j} = 1 \\ \min\{v^{\max}, \mathbf{v}_{i,j-1} + v^{\bullet}_{i,j} - \mathbf{a}_{i,j+1} \cdot e^{\mathrm{over}}_{i,j}\} & \text{otherwise} \end{cases} \tag{16}$$

For each kth preemption point of a task τ_i, we evaluate the source energy level after execution of the basic block $\gamma_{i,k}$ by adding the remaining energy to the energy recovered during execution of $\gamma_{i,k}$, and then substracting the energy consumed by $\gamma_{i,k}$ and by the preemption point if it is active.

Moreover, we consider the saturation of the source energy, *i.e.,* the source energy cannot be greater than $v^{\max}$, and we reset the energy to zero after a preemption point.

For a basic block $\gamma_{i,j}$, we also introduce its energy margin $\mathbf{m}_{i,j}$, which is the difference between the maximum energy level reached and the source's maximum energy since the last active preemption point:

$$\mathbf{m}_{i,1} = \min\{v^{\max}, v^{\max} - \mathbf{v}_{i,1}\} \tag{17}$$

$$\mathbf{m}_{i,j} = \begin{cases} \min\{v^{\max}, v^{\max} - \mathbf{v}_{i,j}\} & \text{if } \mathbf{a}_{i,j} = 1 \\ \min\{\mathbf{m}_{i,j-1}, v^{\max} - \mathbf{v}_{i,j}\} & \text{otherwise} \end{cases} \tag{18}$$

The energy margin is depicted in green in Figure 3. The energy margin represents the maximum harvested energy that can be invested for the execution of the next basic blocs, taking into account the saturation effect of the storage. This means that it is not possible to harvest more energy than this value before the start of the region.

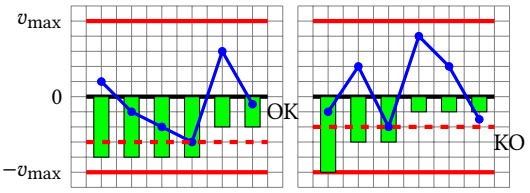

Figure 3: Example of an energy profile for a region starting with an active preemption point. The horizontal axis shows the non-active preemption points within the region, rather than time. The blue curve illustrates the energy level in the storage ($\mathbf{v}_{i,j}$). Each blue point denotes the end of a basic block or a non-active preemption point. The green bars represent the energy margin ($-\mathbf{m}_{i,j}$), with their height indicating the available energy after each basic block. The dashed red line indicates the minimum required energy ($\mathbf{r}_{i,j}$) to ensure execution without energy starvation.

To avoid the starvation state of a region, it is essential to ensure that sufficient energy is available before starting its execution. As discussed earlier, a region can only harvest up to its energy margin. Consequently, the energy consumption at each preemption point must be less than or equal to the available harvestable energy. Figure 3 illustrates this situation. On the right side of the figure, the trajectory represents a scenario where the region will not experience starvation, indicating that adequate energy is stored. Conversely, the left side depicts a scenario where insufficient energy is available to complete the execution, leading to potential starvation. This is captured by the following constraint

$$-\mathbf{v}_{i,j} \leq \mathbf{m}_{i,j} \tag{19}$$

This constraint ensures that there must be an ample energy margin to offset the consumption of a region between two preemption points. If this constraint is not satisfied, it indicates that the energy source cannot guarantee the successful execution of the region.

When this constraint is satisfied, it implies that it is possible to execute the region without starvation. The total amount of energy needed before executing the region must account for all the energy expenditures during the execution of its basic blocks. The minimum energy to be harvested is therefore equal to the lowest energy level that the region can reach (the blue dashed line in Figure 3). This value can be computed recursively. The minimum energy to be harvested, denoted as $\mathbf{r}_{i,j}$, for the region $\Upsilon_{i,j}$ is defined by:

$$\mathbf{r}_{i,n_i} = \min\{0, \mathbf{v}_{i,n_i}\} \tag{20}$$

$$\mathbf{r}_{i,j} = \begin{cases} \min\{0, \mathbf{v}_{i,j}\} & \text{if } \mathbf{a}_{i,j+1} = 1 \\ \min\{\mathbf{v}_{i,j}, \mathbf{r}_{i,j+1}\} & \text{otherwise} \end{cases} : j \in [1..n_i - 1] \tag{21}$$

To begin execution of a region $\Upsilon_{i,j}$, the energy level must reach at least $\mathbf{r}_{i,j}$. The most challenging scenario occurs when the initial energy level is minimal, *i.e.,* zero. In this case, the waiting time $\mathbf{b}_{i,j}$ required before starting the region $\Upsilon_{i,j}$ is:

$$\mathbf{b}_{i,j} = -\frac{1}{w_{load}} \cdot \mathbf{r}_{i,j} : j \in [1..n_i] \tag{22}$$

4.5 ILP solving using Gurobi

In this work, we use the Gurobi Solver to find feasible schedules and optimize our ILP.

In our formulation, some constraints are defined based on specific conditions. Such constraints can be straightforwardly linearized, as shown in [5]. The Gurobi Solver has the capability to automatically handle the linearization of constraints conditioned on a boolean variable[2]. Similarly, Gurobi supports the expression of the global constraint max_, which can be applied to a sets of decision variables to compute the maximum value. Thus, explicit linearization of this constraint is not required when using Gurobi. However, for other solvers, linearization remains a straightforward process.

Note that in equation (6), $\mathbf{b}$ can be directly replaced by $\mathbf{r}$ as follows:

$$\mathbf{z}_{i,j} = \begin{cases} \mathbf{c}_{i,j} - \frac{1}{w_{load}} \cdot \mathbf{r}_{i,j} & \text{if } \mathbf{a}_{i,j} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

Finally, it is possible to optimize an objective function for minimizing preemption costs, or set the objective function to zero, causing the solver to stop at the first feasible solution. The latter approach can reduce solving time.

To optimize the preemption costs, we introduce the cost function defined by:

$$\textbf{Minimize } c(\mathbf{a}) = \sum_{i=1}^{n} \sum_{j=0}^{n_i} \mathbf{a}_{i,j+1} \cdot c_{i,j}^{\text{over}} \tag{24}$$

Note that, to the best of our knowledge, none of the existing limited preemption approaches offer this capability.

Appendix A presents a complete formulation of the problem, including the various equations presented above.

[2]The Gurobi documentation explains that "an indicator constraint $y = f \rightarrow a^T x \leq b$ states that if the binary indicator variable y is equal to f in a given solution, where $f \in \{0, 1\}$, then the linear constraint $a^T x \leq b$ has to be satisfied. On the other hand, if $y \neq f$ (*i.e.,* $y = 1 - f$) then the linear constraint may be violated." https://www.gurobi.com/documentation/9.5/refman/constraints.html

5 RELATED WORK

In this work, we focus on designing energy-neutral real-time systems using limited preemption models. We propose an ILP formulation for selecting preemption points under energy constraints. Consequently, we review related research on the use of ILP for schedulability analysis, as well as studies related to limited preemption. Additionally, we provide an overview of existing research on real-time energy-neutral systems and intermittent computing. To the best of our knowledge, this is the first work that integrates all these concepts.

Significant effort has been made to enhance the efficiency of ILP solvers [5]. Today, ILP-solving libraries such as Gurobi and CPLEX are highly optimized and widely available. These solvers can automatically linearize a broad range of constraints, though automatic linearization is feasible under specific conditions. Modern ILP solvers, especially when used on powerful computing clusters, can solve very large problems within a reasonable time frame despite system complexity. Therefore, solving ILP exactly, rather than approximately, is a practical approach for our problem. Baruah *et al.* have explored ILP formulation to schedulability tests using demand bound function (dbf) approximation [8]. The real-time community has also utilized ILPs for various purposes, including schedulability problems [2, 30], allocation problems [3, 8], and energy optimizations [28].

From the perspective of limited preemption, many studies have focused on controlling preemption costs [6, 9, 13, 16]. Due to space constraints, we review only those most relevant to our work. For a more comprehensive review, see Buttazzo *et al.* [12]. Limited preemption has been addressed through various approaches and models. Preemption Threshold Scheduling (PTS) was introduced in [26], where a task can disable preemption up to a specified priority level (preemption threshold). Preemption is allowed only when a task's priority exceeds the threshold of the preempted task. The limited preemption models used in this work were proposed later. These models can be classified into two categories: the floating preemption point model and the fixed preemption point model. The floating preemption point model [7] defers preemption until the maximum length of the non-preemptive region is reached when a higher priority task arrives. This can reduce the number of preemptions at runtime. The fixed preemption point model [9] selects preemption points in advance during the schedulability analysis phase, allowing preemption only at these predefined points. This approach is similar to ours. Limited preemption methods have been developed for both EDF and Fixed Priority (FP) scheduling. Extending our work to FP would be straightforward and similar to the approach by [27]. However, due to space constraints, this extension is not covered here.

Regarding energy awareness, Dynamic Voltage and Frequency Scaling (DVFS) and Dynamic Power Management (DPM) has been used to control energy consumption [10, 14, 19, 21, 25, 28, 29]. These studies treat energy as an objective, whereas we consider it as a constraint. Research on real-time energy-neutral systems has been conducted over the past two decades.For instance, [4] proposed an optimal scheduler under a harvesting model similar to ours. [24] introduced an optimal Lazy Scheduling Algorithm based on EDF, assuming all tasks consume energy at the same rate using a linear model. Liu *et al.* [23] extended this algorithm to adjust the operating frequency and improve system schedulability. [1] proposed a fixed-priority scheduling algorithm, PF-PASAP, which was the first to consider DVFS in energy harvesting contexts, assuming linear energy consumption and constant power delivery. [15], presented ED-H, an idling and clairvoyant variant of EDF with energy Harvesting capabilities, which is optimal under the assumption that tasks always discharge energy. None of these works account for the impact of preemption on energy consumption as we do. Intermittent system may face power failure during computations, making it challenging to ensure real-time guarantees. This is particularly true for systems powered by unpredictable energy sources like sunlight. To provide real-time guarantees, assumptions about the power supply's evolution are necessary. In [18], Celebi-offline and Celebi-online scheduling algorithms are proposed, introducing charging times before task execution when needed. Only Celebi-offline guarantees real-time guarantees, assuming a priori knowledge of energy availability. Both algorithms aim to minimize charging times while maximizing schedulability. The tasks are either computation or harvesting. The schedulability of the system is validated by simulation over its hyperperiod. In [20], the authors consider an intermittent system whose energy source is characterized by an interval C_c, a period P_c, and a power p: in a period T_c. In [17], the authors present an optimal energy-aware scheduling algorithm for batteryless devices using Mixed Integer Linear Programming (MILP). Their algorithm decomposes the application task into smaller subtasks and determines, based on the available energy, the dec subtasks should be executed.

6 EXPERIMENTATIONS

To evaluate the performance of the ILP formulation, we generated systems with various parameters, such as the number of tasks, processor utilization rate, number of preemption points, etc. The objective is to demonstrate the scalability of this formulation and its applicability to realistically sized systems. The primary metric observed is the time required to compute the optimal solution.

In the second phase of the study, we also examine the impact of the parameter k on overall performance (number of generated constraints, computation time, and accuracy).

We used the Gurobi v10 ILP solver (https://www.gurobi.com) with an Academic Licence. In addition to its high performance, this solver is compatible with numerous programming languages (we used the Python interface) and provides modeling abstractions that simplify the expression of constraints such as min, max, or conditional constraints. The solver automatically reformulates these general constraints into a linearized form.

All experiments were conducted on a MacBook Pro with a 2 GHz Intel Core i5-1038NG7 quad-core processor and 32 GB of memory. The code is available in a repository https://gitlab.univ-nantes.fr/hladik-pe-1/artefact-rtns-2024.

6.1 Task set Generation

For each test, a task set is generated randomly. The system parameters include the number of tasks, processor utilization factor, minimum and maximum number of basic blocks per task, and an

utilization factor representing the overhead induced by preemption points.

For each task, the periods are randomly selected from the set $\{10, 25, 50, 100, 200, 250, 500\}$ and the worst-case execution times are computed using the UUniFast algorithm [11]. Task deadlines are set equal to their periods (although our schedulability analysis remains valid for constrained deadlines).

The number of basic blocks per task is randomly chosen from the range defined by the minimum and maximum number of basic blocks. The worst-case execution time for each basic block is also calculated using the UUniFast algorithm (with the task's execution time as utilization factor) and the best-case times are randomly selected between 20% and 80% of the worst-case execution times. UUniFast is further used to calculate the execution times of each preemption points so that the sum of the execution times corresponds to the overhead utilization factor specified as a parameter.

Energy-related parameters include the maximum energy level of the source and the energy harvesting rate. To ensure that the energy consumption of each basic block remains within the capacity of the storage device's capacity, we adjust the energy consumption by randomly inflating it as necessary. Additionally, we scale down both the basic block execution time and energy consumption by a random factor to calculate the temporal and energy preemption costs.

6.2 Problem Complexity

We conducted an initial experiment to demonstrate how problem complexity increases as a function of the number of preemption points per task. For this experiment, we generated 20 sets of 10 tasks, with a processor utilization rate of 70% and an additional 10% overhead due to preemption points. The parameter we varied was the number of preemption points per task, which ranged from 5 to 30 in increments of 5.

Figure 4 illustrated the number of problem variables and the constraints generated as a function of the number of preemption points. Binary variables represent decision variables, while continuous variables serve as intermediate variables. Linear constraints refer to the "traditional" constraints expressed in linear form, whereas general constraints represent simpler relationships between variables, such as minimum, maximum, absolute value, or logical OR operations. The techniques for translating these general constraints into traditional linear constraints are well-known and are automatically handled by Gurobi. Unsurprisingly, we observe that all these quantities increase linearly with respect to the number of preemption points. Notably, the number of decision variables remains manageable, so the search space does not grow exponentially.

Figure 5 displays the number of solver statuses reached within a 30-second timeout. Four possible statuses can be identified: (1) the solver proved within 30 seconds that the system is infeasible, (2) the solver finds the optimal solution within 30 seconds, (3) the solver times out but finds a suboptimal solution, and (4) the solver times out without finding any solution or before proving that the system is infeasible.

We observe the expected behavior: a small number of preemption points prevents the solver from finding a solution because the basic blocks cause excessively long blocking times, while a larger number

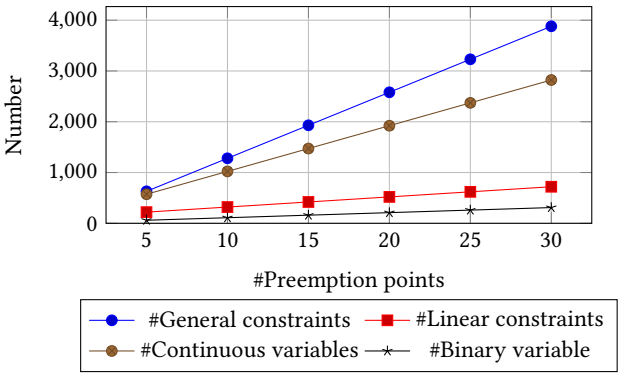

Figure 4: Number of variables and constraints as a function of the number of preemption points for a set of 10 tasks

of preemption points provides greater flexibility in scheduling. It is important to note that in this experiment, the scheduling constraint induced by the overhead of preemption points is quite low (10%), which makes it relatively easy to find solutions. Furthermore, the 30-second time limit for finding a solution often causes the solver to terminate before it can verify the optimality of a solution or even identify an initial solution. However, this time constraint is relatively short given the problem size and was applied here primarily to illustrate overall trends.

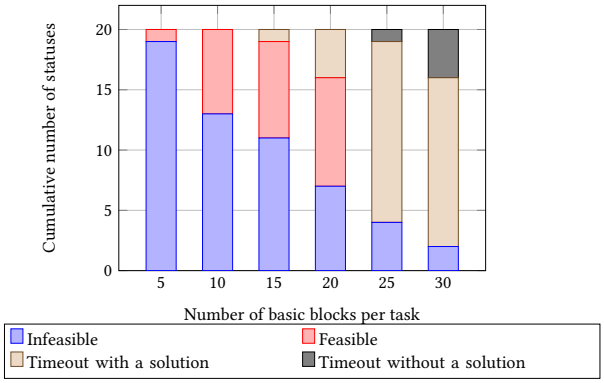

Figure 5: Result statuses for 10 tasks with a 30-second timeout as a function of the number of basic blocks per task

6.3 Influence of preemption point overheads

In this experiment, we vary the utilization rate of preemption point overheads to examine their impact on the overall problem complexity. We used a set of 10 tasks with a total utilization rate of 70%, where each task had 25 preemption points. The overall utilization rate of the preemption points represents the processor load that would be consumed by the overheads of the preemption points if they were all active. The experiment involved varying this rate from 0.1 and 0.6 in increments of 0.1. The time limit for each experiment was set to 300 seconds and we generated 40 sets of tasks for each utilization rate.

Figure 6 shows the distribution of the results across three categories: infeasible solutions, feasible solutions, and timeouts. As expected, increasing the utilization factor of preemption points decreases the feasibility of the problem. We also observe that the problem becomes more difficult to optimize when the utilization factor ranges between 0.3 and 0.4, as indicated by the increased number of timeouts. This is explained by the fact that the problem is constrained by the overheads, but not enough to allow significant pruning of the solution space during exploration.

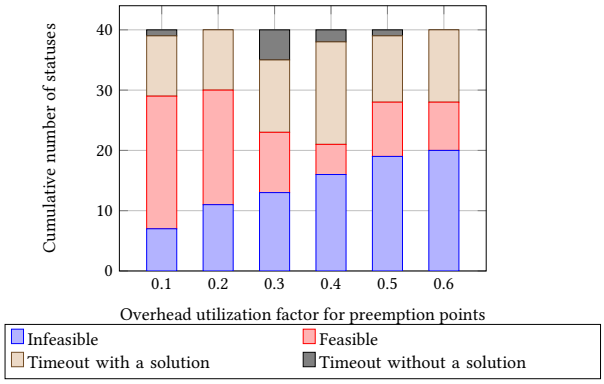

Figure 6: Result statuses for 10 tasks with a 300-second timeout as a function of the overhead of preemption points

The execution times of the solver for producing either a feasible or infeasible solution as a function of the overhead utilization rate are presented in Table 2. We observe that the time required to prove infeasibility is generally short, indicating that in these cases, the problems are highly constrained.

In constrat, the time needed to find the optimal solution is longer, with median values increasing as the overhead utilization rate rises. However, for an overhead utilization rate of 0.6, the median times are lower. This can be attributed to the fact that the problem becomes more constrained at higher overheads, making infeasibility easier to prove and enabling more effective cuts in the search space for feasible solutions.

6.4 Influence of energy constraint

To illustrate a possible application of this method, we use this modeling approach to determine the optimal parameters for sizing the energy source. For a given task system, a binary search method can be employed to find the minimal value of the energy storage capacity, $v^{\max}$, or the lower bound on the amount of power that can be harvested, w^{load}, such that the system is schedulable.

We applied this method with a set of 30 tasks with 40 preemption points per task, a utilization rate of 80%, and an overhead rate of 30%. Our goal was to determine the minimal value of $v^{\max}$ given the amount of power that can be harvested, w^{load}. Note that we used a 30-second timeout for this initial search. It is not necessary to find the optimal solution to verify whether a solution exists for a given $v^{\max}$. Therefor, the search is conducted quickly, and once $v^{\max}$ is determined, a longer timeout can be used to find a potential optimal solution.

Table 2: Solver execution time in second

Overhead	0.1	0.2	0.3	0.4	0.5	0.6
Infeasible problems						
min	0.22	0.06	0.11	0.08	0.14	0.06
1e quartile	3.92	0.30	0.28	0.26	0.34	0.14
median	8.82	6.80	0.45	0.33	0.60	0.27
3d quartile	11.34	8.65	10.27	7.75	4.17	0.34
max	15.08	21.35	46.46	84.20	49.47	23.96
Feasible problems						
min	31.49	23.32	32.11	34.72	30.96	21.70
1e quartile	37.60	33.47	64.65	35.64	70.96	39.57
median	47.35	38.47	74.65	78.85	123.73	49.60
3d quartile	58.76	47.03	86.52	79.67	181.04	114.85
max	222.44	140.63	126.36	106.28	273.42	235.04

6.5 Influence of the scheduling parameter

In this section, we investigate the influence of the parameter k on the expression of the function dbf in Eq. (14). To achieve this, we generate systems of 10 tasks, each having 20 preemption points, a utilization rate of 70%, and an overhead utilization rate of 20%. We vary the parameter k between 1 and 4. The timeout is set to 30 seconds. The task sets remains consistent; only the constraints representing the schedulability are affected by the parameter k.

In terms of model complexity, varying k results in an increase in both the number of continuous variables (which are not decision variables) and the number of linear constraints. Table 3 presents these values.

Table 3: Average number of continuous variables and linear constraints as a function of the parameter k

k	#linear constraints	#continuous variables
1	495	1886
2	515	1916
3	541	1971
4	551	1997

In terms of the impact on solution quality, no difference was observed. The objective value remains consistent for both feasible systems and those exceeding their timeout. Regarding execution time, the difference is not significant. A very slight increase in execution time is noted with increasing values of k, but this remains marginal compared to the variability in time induced by the solver.

Finally, we tested a system composed of 30 tasks, each with 40 basic blocs. A feasible solution was found with a maximum of 150 seconds.

7 CONCLUSION

In this work, we focused on energy-harvesting real-time systems and extended the deferred preemption model to incorporate energy availability constraints.

We employed an ILP representation for selecting preemption points while satisfying both energy and real-time constraints. Our implementation is available online and has been tested under different settings to demonstrate its effectiveness and to highlight the impact of optimality loss when using approximated schedulability analysis compared to exact analysis, versus the benefit in analysis time. We evaluated the performance of the proposed methods using a comprehensive set of synthetic task sets. The results show that our approach is able to handle real-world settings within a very reasonable time frame.

For future work, we plan to incorporated multicore scheduling, for both global and partitioned scheduling. This problem becomes more when considering C-state constraints across all cores. While global scheduling has not yet been addressed, even without energy constraints, even without energy constraints, the partitioned approach offers a more straightforward path. It is feasible to extend our ILP by adding allocation constraints, as the system state space is discrete. We also aim to integrate our scheduler into the Trampoline open-source RTOS to study and validate its practical use and assess its performance on a real target.

REFERENCES

[1] Yasmina Abdeddaïm, Younès Chandarli, and Damien Masson. 2013. The optimality of PFPasap algorithm for fixed-priority energy-harvesting real-time systems. In *Proc. of the IEEE 25th Euromicro Conference on Real-Time Systems (ECRTS)*. 47–56.

[2] Kunal Agrawal, Sanjoy Baruah, and Pontus Ekberg. 2023. Rethinking Tractability for Schedulability Analysis. In *2023 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 1–12.

[3] Ahmad Al Sheikh, Olivier Brun, Maxime Chéramy, and Pierre-Emmanuel Hladik. 2013. Optimal design of virtual links in AFDX networks. *Real-Time Systems* 49, 3 (2013), 308–336.

[4] Andre Allavena and Daniel Mosse. 2001. Scheduling of frame-based embedded systems with rechargeable batteries. In *Workshop on Power Management for Real-time and Embedded systems*.

[5] Mohammad Asghari, Amir M. Fathollahi-Fard, S.M.J. Mirzapour Al-e hashem, and Maxim A. Dulebenets. 2022. Transformation and Linearization Techniques in Optimization: A State-of-the-Art Survey. *Mathematics* 10, 183 (2022).

[6] Sanjoy Baruah. 2005. The limited-preemption uniprocessor scheduling of sporadic task systems. In *17th Euromicro Conference on Real-Time Systems (ECRTS)*. 137–144. https://doi.org/10.1109/ECRTS.2005.32

[7] Sanjoy Baruah. 2005. The limited-preemption uniprocessor scheduling of sporadic task systems. In *Proc. of the 17th Euromicro Conference on Real-Time Systems (ECRTS)*. 137–144.

[8] Sanjoy K Baruah, Vincenzo Bonifaci, Renato Bruni, and Alberto Marchetti-Spaccamela. 2019. ILP models for the allocation of recurrent workloads upon heterogeneous multiprocessors. *Journal of Scheduling* 22 (2019), 195–209.

[9] Marko Bertogna, Orges Xhani, Mauro Marinoni, Francesco Esposito, and Giorgio Buttazzo. 2011. Optimal selection of preemption points to minimize preemption overhead. In *Proc. of the 23rd Euromicro Conference on Real-Time Systems (ECRTS)*. 217–227.

[10] Enrico Bini, Giorgio Buttazzo, and Giuseppe Lipari. 2005. Speed modulation in energy-aware real-time systems. In *Proc. of the 17th Euromicro Conference on Real-Time Systems (ECRTS)*. 3–10.

[11] Enrico Bini and Giorgio C. Buttazzo. 2005. Measuring the Performance of Schedulability Tests. *Real-Time Systems* 30, 1 (2005), 129–154. https://doi.org/10.1007/s11241-005-0507-9

[12] Giorgio C Buttazzo, Marko Bertogna, and Gang Yao. 2012. Limited preemptive scheduling for real-time systems. a survey. *IEEE transactions on Industrial Informatics* 9, 1 (2012), 3–15.

[13] Bipasa Chattopadhyay and Sanjoy Baruah. 2014. Limited-preemption scheduling on multiprocessors. In *Proc. of the 22nd International Conference on Real-Time Networks and Systems (RTNS)*. 225–234.

[14] Jian-Jia Chen and Chin-Fu Kuo. 2007. Energy-Efficient Scheduling for Real-Time Systems on Dynamic Voltage Scaling (DVS) Platforms. In *Proc. of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. 28–38.

[15] Maryline Chetto. 2014. Optimal scheduling for real-time jobs in energy harvesting computing systems. *IEEE Transactions on Emerging Topics in Computing* 2, 2 (2014), 122–133.

[16] Robert I Davis, Alan Burns, Jose Marinho, Vincent Nelis, Stefan M Petters, and Marko Bertogna. 2015. Global and partitioned multiprocessor fixed priority scheduling with deferred preemption. *ACM Transactions on Embedded Computing Systems (TECS)* 14, 3 (2015), 1–28.

[17] Carmen Delgado and Jeroen Famaey. 2021. Optimal energy-aware task scheduling for batteryless IoT devices. *IEEE Transactions on Emerging Topics in Computing* 10, 3 (2021), 1374–1387.

[18] Bashima Islam and Shahriar Nirjon. 2020. Scheduling Computational and Energy Harvesting Tasks in Deadline-Aware Intermittent Systems. In *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. 95–109. https://doi.org/10.1109/RTAS48715.2020.00-14

[19] Ravindra Jejurikar, Cristiano Pereira, and Rajesh Gupta. 2004. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proc. of the 41st ACM annual Design Automation Conference*. 275–280.

[20] Mohsen Karimi, Hyunjong Choi, Yidi Wang, Yecheng Xiang, and Hyoseung Kim. 2021. Real-Time Task Scheduling on Intermittently Powered Batteryless Devices. *IEEE Internet of Things Journal* 8, 17 (2021), 13328–13342. https://doi.org/10.1109/JIOT.2021.3065947

[21] Ying Li, Jianwei Niu, Meikang Qiu, and Xiang Long. 2015. Optimizing Tasks Assignment on Heterogeneous Multi-core Real-Time Systems with Minimum Energy. In *Proc. of the IEEE 17th International Conference on High Performance Computing and Communications, IEEE 7th International Symposium on Cyberspace Safety and Security, and IEEE 12th International Conference on Embedded Software and Systems*. 577–582.

[22] Haining Liu, Ijaz Haider Naqvi, Fajia Li, Chengliang Liu, Neda Shafiei, Yulong Li, and Michael Pecht. 2020. An analytical model for the CC-CV charge of Li-ion batteries with application to degradation analysis. *Journal of Energy Storage* 29 (2020), 101342.

[23] Shaobo Liu, Qing Wu, and Qinru Qiu. 2009. An adaptive scheduling and voltage/frequency selection algorithm for real-time energy harvesting systems. In *Proceedings of the 46th Annual Design Automation Conference*. 782–787.

[24] Clemens Moser, Davide Brunelli, Lothar Thiele, and Luca Benini. 2007. Real-time scheduling for energy harvesting sensor nodes. *Real-Time Systems* 37 (2007), 233–260.

[25] Eduardo Valentin, Mario Salvatierra, Rosiane de Freitas, and Raimundo Barreto. 2015. Response time schedulability analysis for hard real-time systems accounting DVFS latency on heterogeneous cluster-based platform. In *Proc. of the 25th IEEE International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*. 1–8.

[26] Yun Wang and Manas Saksena. 1999. Scheduling fixed-priority tasks with preemption threshold. In *Proc. of the 6th IEEE International Conference on Real-Time Computing Systems and Applications (RTCSA)*. 328–335.

[27] Gang Yao, Giorgio Buttazzo, and Marko Bertogna. 2009. Bounding the maximum length of non-preemptive regions under fixed priority scheduling. In *Proc. of the 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. 351–360.

[28] Houssam-Eddine Zahaf, Abou El Hassen Benyamina, Richard Olejnik, and Giuseppe Lipari. 2017. Energy-efficient scheduling for moldable real-time tasks on heterogeneous computing platforms. *Journal of Systems Architecture* 74 (2017), 46–60. https://doi.org/10.1016/j.sysarc.2017.01.002

[29] Houssam-Eddine Zahaf, Giuseppe Lipari, Marko Bertogna, and Pierre Boulet. 2019. The Parallel Multi-Mode Digraph Task Model for Energy-Aware Real-Time Heterogeneous Multi-Core Systems. *IEEE Trans. Comput.* 68, 10 (2019), 1511–1524. https://doi.org/10.1109/TC.2019.2909886

[30] Houssam-Eddine ZAHAF, Giuseppe Lipari, Smail Niar, and Abou El Hassan Benyamina. 2020. Preemption-Aware Allocation, Deadline Assignment for Conditional DAGs on Partitioned EDF. In *2020 IEEE 26th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. 1–10. https://doi.org/10.1109/RTCSA50079.2020.9203643

A COMPLETE ILP FORMULATION

The problem involves selecting active preemption points for the different tasks while minimizing the overhead caused by the preemption points. This problem can be formulated as an ILP problem.

The decision variables are the $\mathbf{a}$ binary variables:

$$\mathbf{a}_{i,j} \in \mathbb{B} \qquad : i \in [1..n], j \in [1..n_i + 1]$$

Other variables $\mathbf{v}$, $\mathbf{m}$, $\mathbf{b}$, $\mathbf{B}$, etc. are intermediate variables used only to represent constraints.

The optimization problem is formulated as:

$$\textbf{minimize } c(\mathbf{a}) = \sum_{i=1}^{n}\sum_{j=0}^{n_i} \mathbf{a}_{i,j+1} \cdot c_{i,j}^{\text{over}}$$

$$\text{s. t.}$$

$$i \in [1..n]:$$

$$\mathbf{a}_{i,1} = 1$$

$$\mathbf{a}_{i,n_i+1} = 1$$

$$i \in [1..n], j \in [1..n_i]:$$

$$\mathbf{a}_{i,j+1} = 1 \Rightarrow \mathbf{c}_{i,j} = c_{i,j}^{\text{worst}} + c_{i,j}^{\text{over}}$$

$$i \in [1..n], j \in [1..n_i - 1]:$$

$$\mathbf{a}_{i,j+1} = 0 \Rightarrow \mathbf{c}_{i,j} = c_{i,j}^{\text{worst}} + \mathbf{c}_{i,j+1}$$

$$i \in [1..n], j \in [1..n_i]:$$

$$\mathbf{a}_{i,j} = 1 \Rightarrow \mathbf{z}_{i,j} = \mathbf{c}_{i,j} - \frac{1}{w_{load}} \cdot \mathbf{r}_{i,j}$$

$$i \in [1..n], j \in [1..n_i]:$$

$$\mathbf{a}_{i,j} = 0 \Rightarrow \mathbf{z}_{i,j} = 0$$

$$i \in [1..n]:$$

$$\mathbf{w}_i = \sum_{j=1}^{n_i} \mathbf{z}_{i,j}$$

$$\mathbf{q}_i = \max_{j=1}^{n_i} \{\mathbf{z}_{i,j}\}$$

$$\mathbf{B}_n = 0$$

$$i \in [1..n-1]:$$

$$\mathbf{B}_i = \max_{i<k\le n} \{\mathbf{q}_k\}$$

$$i \in [1..n], j \in [1, n_i]:$$

$$\mathbf{a}_{i,j} = 1 \Rightarrow \mathbf{v}_{i,j} = \min\{v^{\max}, v_{i,j}^{\bullet} - \mathbf{a}_{i,j+1} \cdot e_{i,j}^{\text{over}}\}$$

$$i \in [1..n], j \in [2, n_i]:$$

$$\mathbf{a}_{i,j} = 0 \Rightarrow \mathbf{v}_{i,j} = \min\{v^{\max}, \mathbf{v}_{i,j-1} + v_{i,j}^{\bullet} - \mathbf{a}_{i,j+1} \cdot e_{i,j}^{\text{over}}\}$$

$$i \in [1..n], j \in [1..n_i]:$$

$$\mathbf{a}_{i,j} = 1 \Rightarrow \mathbf{m}_{i,j} = \min\{v^{\max}, v^{\max} - \mathbf{v}_{i,j}\}$$

$$i \in [1..n], j \in [2..n_i]:$$

$$\mathbf{a}_{i,j} = 0 \Rightarrow \mathbf{m}_{i,j} = \min\{\mathbf{m}_{i,j-1}, v^{\max} - \mathbf{v}_{i,j}\}$$

$$i \in [1..n], j \in [1..n_i]:$$

$$\mathbf{a}_{i,j+1} = 1 \Rightarrow \mathbf{r}_{i,j} = \min\{0, \mathbf{v}_{i,j}\}$$

$$i \in [1..n], j \in [1..n_i - 1]:$$

$$\mathbf{a}_{i,j+1} = 0 \Rightarrow \mathbf{r}_{i,j} = \min\{\mathbf{v}_{i,j}, \mathbf{r}_{i,j+1}\}$$

$$i \in [1..n], j \in [1..n_i]:$$

$$-\mathbf{v}_{i,j} \le \mathbf{m}_{i,j}$$

$$i \in [1..n], t \in \mathcal{S}_k, t \le (k-1)p_i + d_i:$$

$$\text{dbf}_{\text{i}}^{(\text{k})}(\text{t}) = \left(1 + \left\lfloor \frac{t - d_i}{p_i} \right\rfloor\right) \mathbf{w}_i$$

$$i \in [1..n], t \in \mathcal{S}_k, t > (k-1)p_i + d_i:$$

$$\text{dbf}_{\text{i}}^{(\text{k})}(\text{t}) = \left(1 + \frac{t - d_i}{p_i}\right) \mathbf{w}_i$$

$$i \in [1..n], t \in \mathcal{S}_k:$$

$$\mathbf{B}_i \le t - \sum_{j=1}^{n} \text{dbf}_{\text{j}}^{(\text{k})}(\text{t})$$