Favorable Adjustment of Periods for Reduced Hyperperiods in Real-Time Systems

Dominic Oehlert
dominic.oehlert@tuhh.de
Hamburg University of Technology
Hamburg, Germany

Arno Luppold
arno.luppold@tuhh.de
Hamburg University of Technology
Hamburg, Germany

Heiko Falk
heiko.falk@tuhh.de
Hamburg University of Technology
Hamburg, Germany

ABSTRACT

The hyperperiod defines the time span after which the temporal behavior of a periodical real-time system repeats. It is the key property which determines the complexity of both analysis and exhaustive simulation of a given system. Unfortunately, the hyperperiod may easily become very large. We introduce an ILP-based approach to modify the periods in a task set according to user constraints to retrieve an optimal solution for a drastically reduced hyperperiod.

CCS CONCEPTS

• **Computer systems organization** → *Real-time systems*; • **Mathematics of computing** → *Integer programming*.

KEYWORDS

hyperperiod, real-time, integer linear programming

ACM Reference Format:
Dominic Oehlert, Arno Luppold, and Heiko Falk. 2019. Favorable Adjustment of Periods for Reduced Hyperperiods in Real-Time Systems. In *22nd International Workshop on Software and Compilers for Embedded Systems (SCOPES '19), May 27–28, 2019, Sankt Goar, Germany*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3323439.3323975

1 INTRODUCTION

In a real-time system, tasks are usually executed not only once but on a regular basis. It is important to know after which time interval the execution pattern of the system will repeat.

This time interval is defined by the hyperperiod, which is the least common multiple of all activation periods over all tasks which are present in a given system. Unfortunately, the hyperperiod may subsequently easily become very large. Depening on the schduling algorithm of the system, this leads to a high computational demand for the schedulability analysis. When the system behavior is simulated, large hyperperiods lead to accordingly high simulation times regardless of the underlying scheduling algorithm.

Fortunately, even minor adjustments to tasks' execution periods may lead to a significant reduction of the system's hyperperiod. Of course, when tightening activation periods (either real or for the analysis only), the system's schedulability might slightly decrease. On the other hand, when loosening the periods, the system might no longer work correctly in its physical environment. However, slight tweaks within a fixed range may often be acceptable in real-world systems. In this paper, we propose an ILP-based approach in order to tackle two different scenarios:

1) In the first scenario, we assume to have an existing system with a given task set. Each task has a corresponding worst-case execution time and activation pattern. In order to ease schedulability analysis, system simulation, and to minimize the required memory footprint in case of using an offline scheduler, the aim is to minimize the hyperperiod while adjusting the individual periods as little as possible.

2) The second scenario is set in the domain of synthetic task generation. We assume to have a synthetically generated periodic task set. The utilization of each task was determined by using an appropriate algorithm (e.g., UUniFast [2]). The aim is to obtain a hyperperiod below a certain threshold to allow for a feasible schedulability analysis while ensuring a bound for the maximum deviation for each task utilization, as well as for the overall utilization.

SCOPES '19, May 27–28, 2019, Sankt Goar, Germany

ACM ISBN 978-1-4503-6762-2/19/05…$15.00
https://doi.org/10.1145/3323439.3323975

2 RELATED WORK

Finding a set of suitable periods has been researched under different aspects. Xu [7] presented an algorithm which adjusts given periods such that the hyperperiod of the adjusted periods will be provably always lower than a certain threshold. The algorithm uses highly composite numbers as a an upper hyperperiod threshold and adjusts each period to a nearest multiple of prime factors. To ensure safety, the algorithm only reduces the lengths of periods, but does not enlarge them.

Brocal et al. [3] presented a heuristic to minimize the hyperperiod of a given set of periods, whereas each period is allowed to deviate by a user-definable amount from the original value. They key idea is not to exhaustively calculate all possible hyperperiods of the set of allowed ranges of periods, but only for a chosen subset. The approach is referred as a heuristic by the authors, as it does not return the minimum hyperperiod for systems with a very large minimum hyperperiod (greater than a long long integer).

Nasri and Fohler [5] showed an efficient method to find periods for a given set of period ranges while maintaining a system load smaller than or equal to 100 %. However, their approach is limited to harmonic periods only.

3 DEFINITIONS AND NOTATIONS

In the following, we assume a task set γ consisting of N tasks τ_i, $i = 0, \ldots, N-1$. Each task has a known Worst-Case Execution Time (WCET) C_i and an initial period T_i. The adjusted period of each task τ_i is denoted by $\tilde{t}_i$.

Our approach is not limited to strictly-periodical systems. Whether each task has exactly one period associated to it, or whether it is triggered by a more complex activation pattern composed of multiple basic periods is not critical for applying the presented approach. However, associating one dedicated period T_i to each task τ_i eases both notations, applicability to common task set generators like UUnifast and comparison to previous work.

Capital letters like, e.g., T_i are used to express *constants* in the upcoming ILP model. ILP *variables* are denoted by lower-case letters in any formulas, e.g., $\tilde{t}_i$.

4 OPTIMAL PERIODS

Section 4.1 discusses our ILP model to determine the minimal hyperperiod with restricted period deviations. Section 4.2 shows how the ILP model may be used in order to find a constrained hyperperiod while guaranteeing a pre-defined system utilization for each task.

4.1 Minimal Hyperperiod

For the first scenario, we state the problem as follows:

$$\min \ \mathrm{lcm}\left(\tilde{t}_0, \tilde{t}_1, ..., \tilde{t}_{N-1}\right), \tag{1}$$

$$\tilde{t}_i \in [\lceil T_i \cdot (1-E) \rceil, T_i], \ 0 \leq i < N \tag{2}$$

As Eq. (1) states, our aim is to find a set of periods to minimize their least common multiple. E in Eq. (2) is a user-given real constant in the range of [0, 1], defining the maximum relative deviation allowed per period. An adjusted period $\tilde{t}_i$ is not allowed to be greater than its original period T_i to ensure the system's functionality. We assume that if a task with a period of T_i fulfills its functional requirements inside a system, it will still do so for an adjusted period $\tilde{t}_i \leq T_i$. Using this bound on the maximum period length deviation, we can also derive a maximum deviation from the original system load.

$$\Delta U = \frac{\tilde{U} - U}{U} = \left(\sum \frac{C_i}{(1-E) \cdot T_i} - \sum \frac{C_i}{T_i} \right) \cdot U^{-1} \tag{3}$$

$$= \left(\frac{1}{1-E} \cdot \sum \frac{C_i}{T_i} - \sum \frac{C_i}{T_i} \right) \cdot U^{-1} \tag{4}$$

$$= \left(\frac{1}{1-E} \cdot U - U \right) \cdot U^{-1} = \frac{E}{1-E} \tag{5}$$

U is the original system load, $\tilde{U}$ the maximum system load with adjusted periods and ΔU the maximum relative deviation from the original system load.

In the following, we present an ILP formulation to find the optimal periods to reach a minimal hyperperiod, while ensuring a user-tunable maximum deviation from the original periods. For each original period T_i, an ILP variable $\tilde{t}_i$ is inserted into the ILP:

$$(1-E) \cdot T_i \leq \tilde{t}_i \leq T_i \tag{6}$$

Subsequently, we set up a constraint for each task τ_i to find a common multiple of all periods $\tilde{t}_i$.

$$\tilde{t}_i \cdot f_i = g, \ 0 \leq i < N \tag{7}$$

$$f_i > 0, \ 0 \leq i < N \tag{8}$$

The ILP variable f_i represents any integer factor which is multiplied with the period $\tilde{t}_i$ to achieve a common multiple g of all periods. The variable g is bound to integer values as well.

Obviously, Eq. (7) is not linear, as two ILP variables are multiplied. Yet, we show in the following how the multiplication of two ILP variables can be described efficiently using only linear terms.

Unsigned Multiplication inside an ILP: The following principles were introduced first by Furmanek [1]. The key idea is to execute the multiplication using only binary values. We will start with the multiplication of two unsigned ILP variables a and b.

$$y = a \cdot b \tag{9}$$

Subsequently, the variable b is decomposed to the base 2 as shown by Luppold et al. [4]

$$b = \sum_{i=0}^{L-1} b_i \cdot 2^i \tag{10}$$

The variables b_i are bound to binary values and represent the single bits of b. The constant L is derived by the maximum number of bits required to represent b.

$$L = \left\lceil \log_2 \left(\hat{B} \right) \right\rceil \tag{11}$$

$\hat{B}$ is the maximum allowed value of the integer variable b. Finally, the actual multiplication is expressed by using a min-function and a sufficiently large constant Z, such that $Z \geq \hat{A}$, where $\hat{A}$ is the maximum allowed value of variable a.

$$y = \sum_{i=0}^{L-1} a \cdot b_i \cdot 2^i \tag{12}$$

$$= \sum_{i=0}^{L-1} \min\left(a, b_i \cdot Z\right) \cdot 2^i \tag{13}$$

In case b_i is zero, the min-function will result in 0. Otherwise, the min-function will return a since Z is bound to be always greater or equal to any value of a.

The min-function itself can be described using regular ILP constraints as described by Oehlert et al. [6]. Furmanek also describes the needed extension for multiplying signed integers. Yet, these are not required in this work as the periods will be always positive.

Using the previously described principles, the multiplication in Eq. (7) can be described in a linear fashion. If the period variable $\tilde{t}_i$ from Eq. (7) is chosen as the multiplier (cf. a in Eq. (9)), T_i can be used as a safe value for the large constant Z, as Eq. (6) ensures that $\tilde{t}_i \leq T_i$ always holds.

In order to find the least common multiple (the hyperperiod) of all periods subject to the given constraints, we simply insert the objective term to minimize the common multiple g.

$$\min : g \tag{14}$$

We choose to describe the optimization problem using ILP instead of quadratic constrained programming (QCP) to avoid convexity-requirement issues of solvers. It turned out that both state-of-the-art optimizers CPLEX and Gurobi are not capable of solving the presented set of constraints (without re-formulating Eq. (7)), as they do not fulfill their QCP solver's requirements. Yet, both are capable of solving the ILP problem using the presented re-formulations.

4.2 Limited Hyperperiod With Limited Deviation

For the second scenario, we want to achieve a hyperperiod below a given threshold while ensuring a bound for the maximum deviation for each task's utilization and for the overall utilization. This is desirable as randomly chosen periods easily result in an extremely large hyperperiod, making a schedulability analysis technically infeasible. While the periods should be adjusted in a way that the hyperperiod decreases to a value which makes a schedulability analysis feasible, the utilization per period should stay close to the original one.

The approach by Xu [7] is able to limit the maximum utilization deviation by applying Eq. (5) (as the relative period deviation is limited). This results in the fact that also each task's utilization is limited relatively by ΔU. Xu presents different sets of prime numbers and exponents which correspond to the following set of maximum period deviations: $E_{Xu} = \{0.1, 0.14, 0.25, 0.5\}$. By applying Eq. (5), this results in the following maximum relative utilization deviations: 11%, 16%, 33% and 100%. Yet, in case the user wants to ensure that the maximum utilization deviation is less than 11%, or to choose a value in between, Xu's approach cannot be easily adapted.

In the following, we present an ILP formulation to find a set of periods such that the hyperperiod is lower than or equal to a given threshold, while guaranteeing a maximum, user-definable relative deviation for each partial task utilization, as well as for the overall system's utilization. First up, we define an ILP variable $\tilde{t}_i$ for each original Period T_i.

$$(1 - E_S) \cdot T_i \leq \tilde{t}_i \leq (1 + E_S) \cdot T_i \tag{15}$$

The constant E_S is calculated upfront using the following equation:

$$E_S = \frac{\Delta U}{\Delta U + 1} \tag{16}$$

Eq. (16) is simply Eq. (5) re-phrased. ΔU can be chosen by the user and determines the maximum relative utilization deviation (per task and overall).

In analogy to Section 4.1, we insert the following constraints for each period to calculate a common multiple.

$$\tilde{t}_i \cdot f_i = g \tag{17}$$

$$f_i > 0 \tag{18}$$

The multiplication in Eq. (17) is described in a linear fashion as shown in Eqs. (9) to (13). We enforce the common multiple to be lower than or equal to a given threshold H.

$$g \leq H \tag{19}$$

As we are not interested in actually finding a minimum or maximum value, rather than in simply finding a suitable solution under the given constraints, no optimization objective needs to be added.

5 EVALUATION

All experiments presented in the following sections were performed on an Intel Xeon Server and the ILPs were solved using Gurobi 8.0.0 (multithreaded). We evaluate task sets ranging from 3 to 60 tasks. The tasks' original period lengths are determined using a uniform random distribution ranging from 10 µs to 100 000 µs. The actual time unit of µs is chosen arbitrarily as it does not have any influence on the model. To avoid statistical spikes, each experiment is carried out 20 times. Each data point represents the median among these 20 runs.

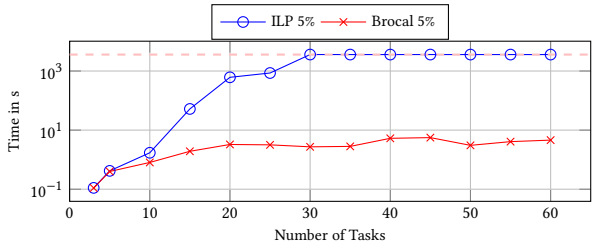

Figure 1: Comparison of the execution times of the ILP model and the Brocal approach for a maximum period deviation of $E = 0.05$.

5.1 Finding The Minimal Hyperperiod

We compare the runtimes of our ILP-based approach against Brocal et al.'s approach [3] to find the minimum hyperperiod possible. The heuristic by Brocal et al. [3] requires to calculate all possible hyperperiods for a subset of M tasks of the whole task set as an initial step. No evaluation or suggestion for this parameter M is given in the paper. We evaluated this parameter in order to achieve a fair comparison of the runtimes with task sets ranging from 5 to 60 tasks and a maximum lower period deviation of $E = 0.1$ for $M = \{1, 2, 3\}$. Results revealed that the heuristic consistently required the lowest runtimes for $M = 1$. We therefore choose to set $M = 1$ for all upcoming experiments.

Fig. 1 shows the comparison of the execution times for finding the minimum hyperperiod when allowing a maximum period deviation of $E = 0.05$ for the ILP model and the Brocal approach. The ILP execution times include the solving time and the time required to set up the ILP formulation. Similarly, the heuristic's runtimes include the complete runtime as well. The overall timeout was set to 1 h, which is indicated as the dashed horizontal line. It can be seen, that as expected, the execution times based on the ILP model show an exponential growth with regard to the number of tasks. Yet, the approach by Brocal does not show a significant increase in runtime for a task set ≥ 20 tasks. For smaller systems (number of tasks < 10), the ILP model finds the minimum hyperperiod in a comparable time as the Brocal approach. For systems with a larger number of tasks, the heuristic by Brocal performs better with the given parameters. For systems with a number of tasks ≥ 30, the ILP always took longer than 1 h and was therefore canceled due to timeout.

The results for the same comparison, but now using a max. period deviation of $E = 0.1$ are depicted in Fig. 2. The median execution time of Brocal's approach with a max. period deviation of $E = 10\%$ and $M = 1$ is 21% lower than with a max. period deviation of only $E = 5\%$. Most likely this is caused by the lower hyperperiods achievable by allowing a greater range. Therefore, the heuristic can return earlier and compensates the greater interval ranges, which otherwise would lead to a greater number of comparisons to be done.

Simultaneously, the ILP model shows significantly slower execution time growth with the larger period deviation allowed. For $E = 0.1$, the ILP approach outperforms the heuristic for systems with a number of tasks < 20.

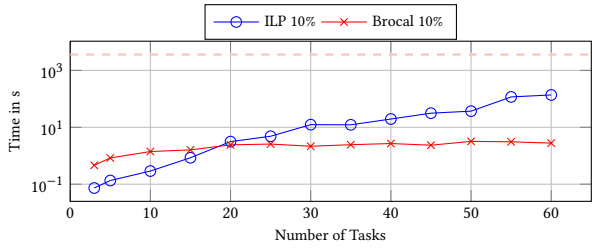

Figure 2: Comparison of the execution times of the ILP model and the Brocal approach for a maximum period deviation of $E = 0.1$.

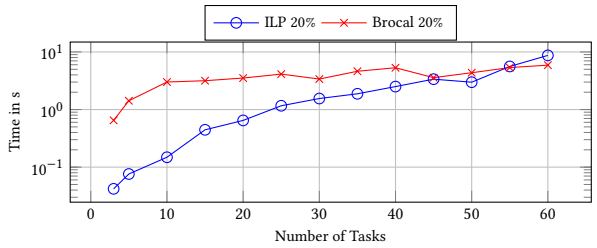

Figure 3: Comparison of the execution times of the ILP model and the Brocal approach for a maximum period deviation of $E = 0.2$.

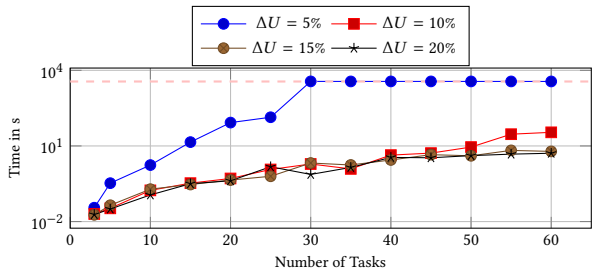

Figure 4: Comparison of the execution times of the ILP-based approach for favorable hyperperiods below a given threshold H for different ΔU values.

Fig. 3 shows the results for a maximum period deviation of $E = 0.2$. Here, the ILP-based approach outperforms the heuristic significantly for systems with a number of tasks < 55. The median runtime of the approach by Brocal is 1.84 times greater for $E = 0.2$ in comparison to $E = 0.05$. In this case, the smaller hyperperiods achievable seem not to outweigh the larger period interval ranges anymore.

In general, it can be observed that our ILP-based approach to find the minimal hyperperiod performs better for smaller task sets or periods with larger allowed ranges than the current state of the art. And it provably finds the minimal hyperperiod, in contrast to the other heuristics.

5.2 Limited Hyperperiod

Fig. 4 shows the evaluation results for the second scenario described in Section 4.2. The ILP model is set up to find any suitable set of periods out of a set of period ranges such that the resulting hyperperiod is less than or equal to a user-definable threshold H. Simultaneously, it guarantees that the adjusted overall system utilization, as well as each adjusted task's utilization, only deviates by an adjustable maximum of ΔU (relatively to the original utilizations).

We evaluate maximum relative utilization deviations $\Delta U = \{0.05, 0.1, 0.15, 0.2\}$. The corresponding period range limits are calculated using Eq. (15) based on the corresponding value of ΔU. The threshold H was set to 1 663 200 µs for $\Delta U = \{0.1, 0.15, 0.2\}$ and to 465 585 120 µs for $\Delta U = 0.05$. These values correspond to limits derived from Xu [7].

For $\Delta U = 0.05$, Fig. 4 shows that the approach only finds a valid solution inside the given timeout limit 1 h for task sets < 30 tasks. For small task sets (< 10), a valid solution is found under 1 s.

All other evaluated values of maximum utilization deviation ΔU show a significantly lower increase in execution time over increasing number of tasks. Overall, the execution times for $\Delta U = \{0.1, 0.15, 0.2\}$ are very similar for almost all evaluated task sets. For task sets below 30 tasks, a valid solution can be found in less than or around a second. Given a set of 60 tasks and maximum utilization deviation of $\Delta U = 0.15$ or $\Delta U = 0.2$, a solution is found in average in 5 s to 6 s, whereas limiting the maximum utilization deviation ΔU to 0.1 results in an average solving time of 35 s.

6 CONCLUSION

In this paper, we presented two novel approaches to find favorable periods for a suitable hyperperiod in a real-time system. The first approach describes an ILP model to find the minimal hyperperiod possible given a set of periods and a maximum period deviation. We showed that our approach outperforms (in terms of runtime) the current state of the art for certain scenarios (small systems or larger allowed period deviation). Our second approach does not aim at finding a minimal hyperperiod, but rather a set of periods resulting in a hyperperiod below a given threshold while guaranteeing a maximum relative utilization deviation. We evaluated this approach in terms of runtime and the influence of the maximum utilization deviation.

ACKNOWLEDGMENTS

This work received funding from Deutsche Forschungsgemeinschaft (DFG) under grant 380772147. This work is part of a project that has received funding from the European Union Horizon 2020 research and innovation programme under grant agreement No 779882.

REFERENCES

[1] Adam Furmanek. [n. d.]. ILP Part 6 - Faster multiplication. https://blog.adamfurmanek.pl/2015/09/26/ilp-part-6/.
[2] Enrico Bini and Giorgio C. Buttazzo. 2005. Measuring the Performance of Schedulability Tests. *Real-Time Systems* 30, 1 (2005).
[3] Vicent Brocal, Patricia Balbastre, Rafael Ballester, and Ismael Ripoll. 2011. Task period selection to minimize hyperperiod. In *Proceedings of ETFA*.
[4] Arno Luppold, Christina Kittsteiner, and Heiko Falk. 2016. Cache-Aware Instruction SPM Allocation for Hard Real-Time Systems. In *Proceedings of SCOPES*.
[5] Mitra Nasri and Gerhard Fohler. 2015. An Efficient Method for Assigning Harmonic Periods to Hard Real-Time Tasks with Period Ranges. In *Proceedings of ECRTS*.
[6] Dominic Oehlert, Arno Luppold, and Heiko Falk. 2017. Bus-Aware Static Instruction SPM Allocation for Multicore Hard Real-Time Systems. In *Proceedings of ECRTS*.
[7] Jia Xu. 2010. A method for adjusting the periods of periodic processes to reduce the least common multiple of the period lengths in real-time embedded systems. In *Proceedings of IEEE/ASME MESA*.