Optimal Resource Allocation and Periodic Scheduling

Roel W.M. van Os, Marc Geilen, Martijn Hendriks, and Twan Basten
Eindhoven University of Technology
Eindhoven, Netherlands
Email: {r.w.m.v.os, m.c.w.geilen, m.hendriks, a.a.basten}@tue.nl

***Abstract*—Many real-world applications deal with the problem of scheduling repetitive tasks in a periodic fashion. Often, such tasks involve precedence constraints and require sharing a limited set of, possibly heterogeneous, resources to enable their execution. This results in a problem that combines resource allocation and periodic task scheduling, which we refer to as the Allocation and Periodic Scheduling Problem (APSP). This paper proposes two linear-programming models for solving the APSP based on a novel modeling approach. The first model is a monolithic Mixed-Integer Linear Programming (MILP) model and the second one is a MILP model with a Benders decomposition. Both models allow finding an optimal resource allocation and periodic schedule that minimizes the period, i.e., they optimize throughput. In contrast to earlier work on optimally solving the APSP, our approach uses a time bound on considered start times of task executions, meaning that only the repetitions starting within the time bound are considered in the solving process. This effectively constrains the search space for finding optimal schedules. For feasible problem instances, we prove that there exists an optimal schedule in which all tasks execute at least once within this time bound, showing the soundness of the approach. Where earlier models for the APSP are in essence non-linear, our approach results in easier-to-solve linear models. Experiments show that, in comparison to the state of the art, our models are less prone to numerical instabilities and find optimal solutions more often, while being competitive in solve time.**

***Index Terms*—optimization, periodic scheduling, resource allocation, cyclic job scheduling, dataflow graphs, mathematical programming, Benders decomposition**

I. Introduction

Task scheduling problems arise in many domains and are essential for efficient operations management. Traditionally, such scheduling problems are defined over a finite set of tasks within a bounded time horizon, with optimization objectives such as makespan, lateness, or tardiness. More recent problems in operations research involve systems that operate continuously, process unbounded tasks, product flows, or data streams, and exhibit periodic and repetitive behavior. These so-called *periodic scheduling* problems arise in domains such as manufacturing, logistics, and computing. In manufacturing optimization, automated robotic cells are taking over repetitive tasks in mass production systems. Similarly, large retail chains and hospitals face logistic scheduling problems due to the need for periodic inventory replenishment. In embedded computing systems, such as in modern vehicles, tasks related to the processing of sensor data continuously arrive and must be scheduled with real-time constraints. In many applications, the periodic tasks are part of an interconnected system, resulting in task dependencies that impose precedence constraints among the various tasks and task repetitions. This leads to a challenging scheduling problem.

Strict periodicity—where tasks are executed at fixed, regular intervals—is often essential for ensuring temporal predictability and enabling effective resource utilization across system components. To find a feasible strictly periodic schedule, one needs to determine the exact timings of each task and the period at which all tasks recur. As a shorter period directly corresponds to higher throughput, the efficiency of periodic schedules is typically expressed by the length of the period. An optimal strictly periodic schedule is therefore one that minimizes the period while respecting all task dependencies.

Frequently, additional resource allocation challenges arise, where limited resources are shared among the various tasks. Moreover, these resources are often *heterogeneous resources*, meaning that task behavior may be influenced by the specific resource it is assigned to. To find an optimal periodic schedule, one needs to not only consider the timing of the respective tasks but also allocate them efficiently among the available resources. Finding optimal periodic schedules while considering the allocation of resources is not straightforward and is typically NP-hard [36].

This paper addresses the problem of finding the optimal resource allocation and strict-periodic schedule for a set of repetitive tasks with precedence constraints. More specifically, given a set of tasks with precedence constraints that need execution on heterogeneous resources, we aim to find a task-to-resource allocation, start times for each task, and the smallest period duration such that every task repeats strictly periodically in each period. We name this problem the Allocation and Periodic Scheduling Problem (APSP)[1]. From results in [16], it follows that this problem is NP-hard.

The main contribution of this paper is a novel Mixed-Integer Linear Programming (MILP) formulation for solving the APSP, along with two original MILP models that provide optimal solutions, one *monolithic MILP model* and one model with a *Benders decomposition*. Our approach is based on

[1]With APSP, we propose a neutral name and abbrevation to emphasize the domain-independence of the problem. It is commonly known as the flexible cyclic job-shop scheduling problem in operations research and the periodic binding and scheduling of homogeneous synchronous dataflow graphs in embedded computing research.

proving the validity of a bound on the duration of a fragment of an optimal schedule in which all tasks execute at least once (Theorem 1). In this way, we can constrain the search space without losing optimality of the obtained schedule. The time bound makes it straightforward to specify a linear model of the problem, allowing us to avoid the non-linearity and linearization that arises in the current state-of-the-art literature [34, 33]. Benchmarking shows that our bounded models are less sensitive to numerical instabilities. This results in an improvement of 34% in optimal solutions found compared to the state-of-the-art related work, with a competitive solve time.

This paper is organized as follows. Section II introduces an illustrative example. Relevant literature is discussed in Section III. Section IV introduces the APSP and Section V shows how to bound the search for a schedule to a fragment of the schedule without losing optimality. Section VI and Section VII discuss the monolithic MILP model and the Benders decomposition, respectively. Section VIII presents the experimental evaluation. Lastly, in Section IX, we close with conclusions and recommended future work.

II. Illustrative Example

To illustrate our work, we consider a hypothetical real-world-inspired example of a LiDAR-based SLAM (Simultaneous Localization and Mapping) application on a heterogeneous CPU platform. Such applications are common in robotics and autonomous vehicles, where the application periodically constructs 3-dimensional maps of the environment and detects dynamic objects for collision avoidance. The resulting data stream may serve as input to vehicle control applications, which require a predictable real-time input rate.

The application consists of tasks whose execution times depend on the heterogeneous resources to which they are assigned. Modern heterogeneous CPU platforms often combine multiple core types optimized for different workloads, such as general-purpose superscalar processors and digital signal processors (DSPs). All tasks repeat strictly periodically with the same uniform period. Each periodic execution is referred to as a task repetition[2]. Task repetitions are non-preemptive and cannot overlap, meaning that once a repetition starts executing on a resource, it completes before the resource is released for another task repetition.

The goal is to determine the optimal task-to-resource allocation and periodic schedule that minimizes the repetition period of the tasks, thereby maximizing throughput. We treat throughput as an optimization objective, rather than a constraint, to support system dimensioning. Maximizing throughput exposes scheduling bottlenecks and highlights over- or under-dimensioned resources.

The LiDAR-based SLAM application is inspired by [26, 29] and modelled as the task graph in Figure 1. Nodes represent tasks and edges represent precedence constraints.

[2]In real-time scheduling, task repetitions are referred to as jobs. In job-shop literature, the term 'job' has a different meaning though. In dataflow literature, task repetitions are referred to as actor firings. The terminology used here is neutral across these domains.

Task	Execution times per resource (μs)		Description
	r_1	r_2	
t_1	800	950	LiDAR
t_2	19340	12000	Ground segmentation
t_3	6000	9000	Transform points to coordinates
t_4	11000	15000	Remove static obstacles
t_5	4000	2500	Clustering
t_6	5000	3200	Principal component analysis
t_7	9000	7000	Dynamic object filter
t_8	22000	14000	Scan matching
t_9	29200	21050	Pose graph optimization
t_{10}	710	920	GNSS
t_{11}	3500	5200	Global position update

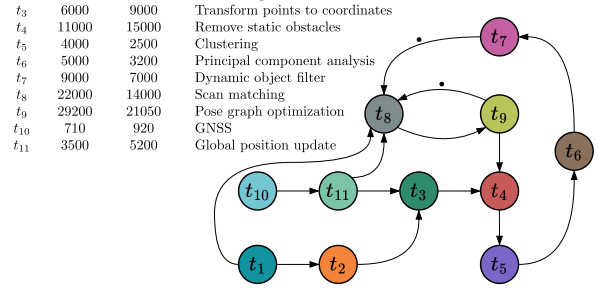

Fig. 1. Task graph of the LiDAR-based SLAM application.

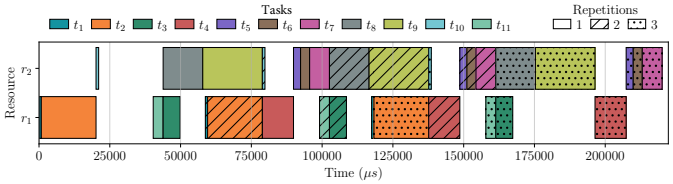

Fig. 2. Optimal periodic schedule of the LiDAR-based SLAM application.

Some dependencies include tokens, shown as black dots, that indicate the dependency depth: a dependency holds between a task repetition and its successor's repetition offset by the specified number of tokens (i.e., n tokens on a dependency from task t_i to task t_j indicates that repetition k of t_i needs to precede repetition $k+n$ of t_j). LiDAR point clouds and GNSS measurements are periodically acquired and fused to estimate sensor poses through scan matching against known static structures. This information is used to optimize a pose graph, which results in a 3d map of the static environment. Dynamic objects are filtered for reliable matching. After ground removal, LiDAR points are transformed into global coordinates and compared against the static map to identify moving objects. These points are clustered and classified to extract information about the moving objects, which is fed back into the SLAM pipeline to prevent dynamic objects from degrading localization and mapping accuracy. Task execution times per resource are estimated as shown in Figure 1. A key point is that data dependencies may be cyclic across task repetitions. The model in Figure 1 has two such cycles, both marked with a single token. Tokens within a cycle in the application graph constrain the pipelining depth of tasks in the cycle: the degree at which task repetitions in such a cycle can be executed unconstrained by their dependencies. For instance, tasks $t_8, t_9, t_4, t_5, t_6, t_7$ cannot be pipelined because the cycle through these tasks contains only a single token.

With the proposed approach, we find an optimal allocation and periodic schedule for this LiDAR-based SLAM pipeline, shown in Figure 2. The period is $58750\ \mu s$, which corresponds to a throughput of roughly 17 Hz. The schedule illustrates the challenges. For example, although task t_2 executes significantly faster on resource r_2, it is assigned to r_1 in the optimal solution to better exploit overall task-level parallelism. Additionally, different task repetitions interleave in the optimal schedule, for instance, repetition 2 of t_2 comes before repe-

tition 1 of t_4. Furthermore, after repetition 1 of task t_2, both processors remain idle for some time. While it may appear that repetition 1 of t_8 could start earlier, its second repetition depends on repetition 1 of t_7. Since we require all tasks to share a uniform period, the execution of t_8 is constrained by dependencies across all repetitions.

III. Related Work

Periodic scheduling is found in many branches of operations research and dates back as far as the 1960s [22]. However, a foundational contribution came from Serafini and Ukovich [36] in 1989. They proposed the first unifying mathematical framework to describe periodic scheduling problems across different domains. Their mathematical model and problem description would be used as the basis in periodic scheduling research for many years to come.

One variant described by Serafini and Ukovich [36] relates to optimal job scheduling problems that are often studied in the industrial engineering domain. These problems revolve around the optimal scheduling of a set of tasks, a job in their terminology, with precedence constraints on a set of resources, often machines. Specifically, Serafini and Ukovich [36] is one of the first to describe a cyclic job-shop scheduling problem, which is a variant of the traditional job-shop scheduling problem with cyclic jobs where all tasks repeat with a uniform period. Various works followed, such as by Roundy [35], who introduced the NP-hard cyclic job-shop scheduling problem with identical jobs. Roundy proposed a custom search algorithm to obtain periodic schedules for this problem. Hanen [15] considered non-identical jobs and proved NP-hardness for the cyclic job-shop scheduling problem. Both the first exact mathematical model and heuristic procedures were provided for effectively solving the problem. Several works by Brucker and Kampmeyer [10, 9] and Bożejko et al. [8] continued the work of Hanen to extensively study heuristic and exact solutions to the cyclic job-shop scheduling problem.

While these studies have been foundational to the periodic scheduling literature, resource allocation is not considered part of the problem. Hence, Hanen and Munier [16] extended their prior work from [15] to include resource allocation and proved NP-hardness of the resulting problem, along with providing a greedy heuristic to solve it. The resources considered in [16] are homogeneous, making the durations of tasks on each resource identical.

Later, Bożejko, Pempera, and Wodecki [7] introduced the *flexible* cyclic job-shop scheduling problem, that is, the cyclic job-shop scheduling problem where each task needs to be allocated to a heterogeneous resource. This is the problem we refer to as the APSP. The problem is NP-hard, as is easily concluded from the NP-hardness result for the homogeneous variant of [16]. It is solved with a simulated annealing heuristic approach that minimizes the period of the schedule. Quinton, Hamaz, and Houssin [34, 33] proposed exact monolithic MILP models and a Benders decomposition to find optimal shortest-period solutions to this flexible cyclic job-shop scheduling problem. The work in [34, 33] is the only work that provides exact approaches to solve the APSP. These methods are therefore the primary references for our work.

Besides in the industrial engineering domain, periodic scheduling is also widely studied in the computer scheduling domain. In this domain, Serafini and Ukovich [36] caught the attention of Korst et al. [19] who complemented the work of [36] by solving a periodic multiprocessor scheduling problem that includes resource (processor) allocation constraints. Their problem variant does not include precedence constraints between tasks and considers homogeneous resources. Similar problems as the one introduced by Korst et al. gained significant importance in the scheduling of time-triggered real-time systems. Here, multiple independent tasks have to be scheduled given some real-time constraints, characterized by distinct periods for each task. Meeting such constraints requires timing predictability, and periodic scheduling proved to be an effective tool to obtain this. Minaeva and Hanzálek [27] surveyed a large number of articles concerning periodic scheduling for time-triggered hard-real-time systems. They show that the majority of the related work in the literature considers homogeneous resource scheduling and assumes no precedence constraints among the tasks. Notable exceptions are the works by Panić et al. [30], Chen et al. [12], Hladík, Minaeva, and Hanzálek [17], Koulouris and Georgiadis [20], and Chen et al. [11]. While these works come close to the APSP, they limit their task model to non-cyclic sequenced tasks, often represented by directed acyclic graphs. This makes it impossible to capture cyclic dependencies between subsequent repetitions of tasks, which we do consider in the APSP.

Another branch of research deals with so-called streaming applications, which have many applications in consumer electronics as well as industrial applications such as in the automotive domain [4]. These streaming applications are often expressed in the Synchronous Dataflow (SDF) model of computation of Lee and Messerschmitt [21], which captures the precedence relations, or data dependencies, among various repetitions of the application. Periodic scheduling of these applications requires finding a uniform period for all task repetitions. Finding effective periodic schedules for these streaming applications on multiprocessor platforms poses an interesting problem as both precedence relations and the resource allocation become an integral part of the scheduling problem. This essentially results in the APSP.

Bonfietti [4] discusses various streaming-based periodic scheduling problems and classifies various types of scheduling approaches. Among them is the static-order approach, which aims to determine a fixed ordering of tasks that can be periodically repeated. State-of-the-art works regarding this method trace back to the works of Stuijk et al. [39] and Moreira, Valente, and Bekooij [28] that propose different heuristics to effectively solve the problem of allocation and scheduling of multiple SDF applications on heterogeneous multiprocessors for maximal throughput. Additionally, various exact techniques have been proposed by Liu et al. [24], Bonfietti et al. [6], and Liu, Gu, and Ye [23] to solve the same problem to optimality. The resulting static-order schedules are

effective for run-time optimization. However, the absence of task timing in the resulting static-order schedules makes them unsuitable to enforce strict periodicity.

Three other scheduling approaches classified in [4] do contain the explicit timing of tasks, namely the blocked-scheduling, unfolding, and modulo-scheduling approaches. Bonfietti also points out that these scheduling approaches are used in various domains. The blocked-scheduling approach from Sriram and Bhattacharyya [37] considers a single scheduling period, which includes a single repetition of all tasks, that can be repeated indefinitely. Optimal schedules may require including different task repetitions within the same period though (as we saw in the example in Section II), making the blocked-scheduling approach potentially sub-optimal. This motivates the unfolding approach, which applies a blocked scheduling over a fixed number of task repetitions, as seen in Parhi and Messerschmitt [32] and Sriram and Bhattacharyya [37]. While potentially allowing more overlap between repetitions within the schedule, the required number of task repetitions to reach an optimal solution is not known in these works. The unfolding or blocking factor, i.e., the number of repetitions to consider, is typically estimated, therefore, sacrificing optimality of the schedule. The modulo-scheduling approach classified in [4] is based on the cyclic job-shop scheduling framework of Hanen [15]. This approach traditionally schedules a single repetition of all tasks. A periodic schedule is then obtained by repeating tasks every μ time units. This μ is called the modulus and can be much smaller than the length of the single-repetition schedule. This allows overlapping executions of subsequent single-repetition schedules (as in Section II), making it a suitable approach for finding optimal periodic schedules, in contrast to other approaches.

The modulo-scheduling approach is a widely adopted method for optimal periodic scheduling, effectively solving various scheduling problems. It is applied in the discussed related works [15, 16], as well as in the primary references for our work [34, 33]. Additionally, Bonfietti et al. [5] discuss a constraint-programming-based modulo-scheduling approach for solving a variant of [16] that includes variable resource capacity. Their model bounds the number of repetitions but does not discuss whether optimality is preserved. The works of Bodin, Munier-Kordon, and Dinechin [3] and Bodin and Kordon [2] develop modulo-scheduling approaches to compute throughput-efficient/optimal *k-periodic* schedules. Unlike the original modulo-scheduling approach—that schedules a single repetition of each task—their techniques allow for a variable number of task executions within each period. The result is a potentially higher throughput, at the expense of strict periodicity. These works do not include resource allocation as part of the problem.

In this paper, we introduce a new model based on the modulo-scheduling approach to find optimal resource allocations and periodic schedules for the APSP. The approach differs from the state-of-the-art approaches in [34, 33]. In Figure 2, we see that a feasible periodic schedule defines a concrete ordering of start times for different task repetitions, consistent with precedence and timing constraints. In [34, 33], inter-repetition ordering relations are modeled by distance variables that determine the relative offset between a given repetition of different tasks. This results in non-linear constraints. In contrast, our approach exploits the time bound we establish in Theorem 1 to exhaustively evaluate all task repetition orderings within a finite time window, thereby avoiding non-linear constraints. This new approach improves numerical stability and finds optimal solutions to more of the benchmarked problems.

IV. Problem definition

We start by constructively defining the elements of the APSP. The essence of the problem is defined by a set of tasks and their interdependencies. A task is a process that periodically repeats. A repetition of a task describes a single occurrence of this task. A task dependency describes a precedence relation between two tasks. Tasks may be pipelined, i.e., dependencies may exist between different repetitions of tasks. Tasks are executed on a set of heterogeneous resources. Task execution times may depend on the resource that executes them, and it may not be possible to execute specific tasks on some of the resources.

An APSP instance is described by a Task and Resource Model (TRM). For notation, the sets of natural and real numbers are denoted by $\mathbb{N}$ and $\mathbb{R}$, respectively. $\mathbb{N}_0$ denotes the natural numbers including 0. Superscripts can specify a range within a set of numbers; $^+$ specifies all positive numbers and the inequality symbol $\leq$ can be used to specify bounds. $\mathbb{R}_0^+$ denotes the non-negative real numbers.

Definition 1 (Task and Resource Model (TRM))**.** *A TRM is a tuple (T, D, R, e), where*

- *T is a finite set of tasks,*
- *$D \subseteq T \times \mathbb{N}_0 \times T$ is a finite set of task dependencies describing precedence relations between tasks and their depth,*
- *R is a finite set of heterogeneous resources,*
- *$e : T \times R \rightharpoonup \mathbb{R}_0^+$ is a partial function describing the execution times of tasks on the resources on which they can be executed.*

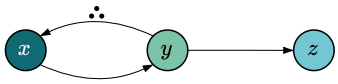

Fig. 3. TRM with three tasks $x, y, z \in T$ and three task dependencies.

We can visualize the task model of a TRM using the graphical representation from the Homogenous Synchronous Dataflow Graph (HSDFG) model of computation of Lee and Messerschmitt [21]. Figure 3 gives an example. A node represents a task in T and an edge represents a task dependency in D. Tokens on an edge indicate depth of a task dependency. Figure 3 shows a simple TRM with tasks x, y, and z. The three initial tokens on the edge from task y to task x indicate that repetition k of task y needs to precede repetition $k+3$ of

task x. A TRM allows a feasible schedule if, and only if, it has a feasible allocation and no cyclic dependencies between the same repetitions of a pair of tasks, i.e., if its graphical representation does not have tokenless cycles. We refer to such TRMs as feasible, and only consider feasible TRMs in the remainder.

A TRM expresses an APSP instance. An APSP solution requires a resource allocation and a schedule. As explained, not all resources may be suitable for a task. For convenience, we define the suitable set of resources for a task t as follows:

$$R_t = \{r \in R \mid e(t, r) \text{ is defined}\}$$

Using R_t, we define the resource allocation for a TRM.

Definition 2 (Resource Allocation)**.** *Let (T, D, R, e) be a TRM. A resource allocation $\alpha : T \to R$ for the TRM specifies for every task $t \in T$ a suitable resource in R_t to which it is allocated. The execution time of task t under allocation α, $e(t, \alpha(t))$, is abbreviated by $e_\alpha(t)$.*

When a TRM and its resource allocation are known, a schedule can be constructed that specifies the start times of each task repetition. A strictly-periodic schedule—henceforth referred to as simply a *periodic schedule*—additionally specifies a period at which task repetitions repeat.

Definition 3 (Periodic Schedule)**.** *Let (T, D, R, e) be a TRM. A* schedule *of the TRM is a function $\sigma : T \times \mathbb{N}_0 \to \mathbb{R}_0^+$ that for each task gives the start times of all of its repetitions. A* periodic *schedule is a pair (σ, μ) where σ is a schedule and $\mu \in \mathbb{N}$ is its period. For all tasks $t \in T$ in a periodic schedule (σ, μ), we require that, for all repetitions $k \in \mathbb{N}_0$, $\sigma(t, k+1) = \sigma(t, k) + \mu$.*

Observe that, for all tasks $t \in T$ in a periodic schedule, if μ and $\sigma(t, 0)$ are given, then we can derive the start times for all succeeding repetitions $\sigma(t, k)$ for all $k \in \mathbb{N}$. Therefore, we introduce the shorthand notation $\sigma(t) = \sigma(t, 0)$ for all $t \in T$ as a sufficient means to characterize a periodic schedule with period μ.

A periodic schedule is only feasible if it adheres to a set of constraints.

Definition 4 (Feasible Periodic Schedule)**.** *Let (T, D, R, e) be a TRM and α its resource allocation. Let $\rho(t)$ denote the completion times of task executions, i.e., $\rho(t) = \sigma(t) + e_\alpha(t)$. The following constraints apply to a feasible periodic schedule (σ, μ):*

- *Task execution adheres to dependencies:*

$$\forall (t_1, n, t_2) \in D : \sigma(t_2) \geq \rho(t_1) - n\mu \tag{1}$$

- *Any repetition of any task t_1 cannot overlap with any repetition of (possibly the same) task t_2 on the same resource. It suffices to disallow overlap of any repetition of t_1 with the first repetition of t_2, and vice versa. As all tasks share a uniform period, the constraint will be repeated in succeeding repetitions. By considering all possible task pairs, the constraint covers both orderings of any two tasks.*

$$\begin{aligned} &\forall k \in \mathbb{N}_0 : \forall t_1, t_2 \in T : \\ &\alpha(t_1) = \alpha(t_2) \wedge (t_1 \neq t_2 \vee k > 0) \\ \implies\ &\rho(t_1) + k\mu \leq \sigma(t_2) \vee \rho(t_2) \leq \sigma(t_1) + k\mu \end{aligned} \tag{2}$$

A periodic schedule (σ, μ) is feasible if and only if all of these constraints are satisfied, and infeasible otherwise. A feasible schedule is optimal if it has the smallest possible period μ.

Example 1. *Figure 4 shows an example of a feasible schedule (σ, μ) for the task model seen in Figure 3. The resource model is described by set $R = \{r_1, r_2\}$. Allocations are specified as $\alpha(x) = r_1$, $\alpha(y) = r_2$, $\alpha(z) = r_1$, with resulting execution times $e_\alpha(x) = 3$, $e_\alpha(y) = 3$, $e_\alpha(z) = 2$. The schedule has period $\mu = 5$, with $\sigma(x) = 0$, $\sigma(y) = 7$, and $\sigma(z) = 13$.* ■

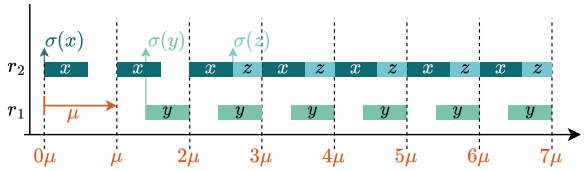

Fig. 4. Feasible schedule of a TRM described by the task model in Figure 3.

Given a feasible TRM with a resource allocation, we know that a feasible periodic schedule exists. However, finding a suitable resource allocation and schedule that minimize the period is not straightforward.

Definition 5 (Allocation and Periodic Scheduling Problem (APSP))**.** *Let (T, D, R, e) be a feasible TRM. The Allocation and Periodic Scheduling Problem is to find a resource allocation α (Definition 2) and a feasible periodic schedule (σ, μ) (Definition 4) that minimize period μ.*

The APSP is known as the binding and scheduling of HSDFGs in embedded computing research. In this context, tasks in the TRM are actors of the HSDFG and task dependencies are the channels between actors. In operations research, the APSP is often called the flexible cyclic job-shop scheduling problem, where resources are typically machines. Dependency depth is then referred to as the height of a dependency.

V. Bounding the Start Times

Periodic schedules do not have a constrained time domain and task repetitions continue indefinitely. When constructing MILP models for the APSP, this poses a problem, as the model can only accommodate finitely many constraints, while the schedule itself is an infinite structure. In our approach, we aim to construct a fragment of an optimal periodic schedule, with finitely many task repetitions, such that it allows a compact, linear constraint specification. The following theorem captures the key result underlying our approach: without loss of optimality, it is possible to bound the duration of a fragment of an optimal schedule in which all tasks execute at least once.

Theorem 1. *Let (T, D, R, e) be a TRM and α a resource allocation for it. If there is a feasible periodic schedule with period μ (Definition 4), then there exists a feasible schedule (σ, μ) such that for all $t \in T$, $\sigma(t) < (|T| + |R|)\mu$.*

Proof. Let (σ', μ) be a feasible periodic schedule. We derive another feasible schedule (σ, μ) by shifting the start times σ' with integer multiples of μ such that σ is as *compact* as possible (i.e., setting the start times as early as possible).

Observation. As all tasks share a uniform period, the no-overlap constraint (2) *holds for all repetitions. Hence, shifting the feasible start times σ' by multiples of μ to either direction does not cause violations of the no-overlap constraint* (2)*. It may violate the task-dependencies constraint* (1)*.*

Guided by this observation, we create σ by only considering shifts of multiples of μ. We thus need to decide in which (integer) *period* of length μ a task should perform its initial repetition.

The *phase* of a task within a period of a given schedule is the offset of task repetitions with respect to the start of the period. It is fixed for a given periodic schedule. Our transformation of σ' into σ preserves task phases. The phase $\hat{\sigma}$ is defined from the given σ' as follows:

$$\hat{\sigma}(t) = \sigma'(t) \mod \mu \tag{3}$$

We determine *start periods* $p : T \to \mathbb{N}_0$ as the period in which task t executes its initial repetition $\sigma(t)$, where $p(t) = i$ denotes the interval $[i\mu,\ i\mu + \mu)$. The actual start time $\sigma(t)$ of the initial repetition of t is then:

$$\sigma(t) = p(t)\mu + \hat{\sigma}(t) \tag{4}$$

To determine p, we translate the task dependencies, (1) in Definition 3, into constraints on p. For $(t_1, n, t_2) \in D$,

$$\begin{aligned} \sigma(t_2) &\geq \sigma(t_1) + e_\alpha(t_1) - n\mu \\ p(t_2)\mu + \hat{\sigma}(t_2) &\geq p(t_1)\mu + \hat{\sigma}(t_1) + e_\alpha(t_1) - n\mu \\ \mu(p(t_2) - p(t_1) + n) &\geq \hat{\sigma}(t_1) - \hat{\sigma}(t_2) + e_\alpha(t_1) \end{aligned} \tag{5}$$

As $p(t_1), p(t_2), n \in \mathbb{N}_0$, to satisfy the task-dependency constraints of (1), we need to require:

$$p(t_2) - p(t_1) + n \geq \left\lceil \frac{\hat{\sigma}(t_1) - \hat{\sigma}(t_2) + e_\alpha(t_1)}{\mu} \right\rceil \tag{6}$$

Because tasks are bound to a resource and executions on a resource cannot overlap, the existence of a feasible periodic schedule (σ', μ) implies that, for all $r \in R$:

$$\sum_{t \in T \ s.t.\ \alpha(t) = r} e_\alpha(t) \leq \mu \tag{7}$$

This gives us a (pessimistic) bound of $e_\alpha(t) \leq \mu$ for all $t \in T$. Since phases $\hat{\sigma}$ fall in the interval $[0, \mu)$, this gives the following bounds on the numerator in (6):

$$-\mu < \hat{\sigma}(t_1) - \hat{\sigma}(t_2) + e_\alpha(t_1) < 2\mu \tag{8}$$

Combining this with (6) gives us the following constraints. For all $(t_1, n, t_2) \in D$,

$$p(t_2) - p(t_1) \geq d(t_1, t_2) \tag{9}$$

where $d(t_1, t_2)$ is defined as follows:

$$d(t_1, t_2) = \left\lceil \frac{\hat{\sigma}(t_1) - \hat{\sigma}(t_2) + e_\alpha(t_1)}{\mu} \right\rceil - n \tag{10}$$

$$= \begin{cases} -n & \text{if } \hat{\sigma}(t_1) - \hat{\sigma}(t_2) + e_\alpha(t_1) \leq 0 \\ 1 - n & \text{if } 0 < \hat{\sigma}(t_1) - \hat{\sigma}(t_2) + e_\alpha(t_1) \leq \mu \\ 2 - n & \text{if } \mu < \hat{\sigma}(t_1) - \hat{\sigma}(t_2) + e_\alpha(t_1) \end{cases} \tag{11}$$

This is the smallest number of periods μ after which a repetition of task t_2 can execute after the same repetition of task t_1 is executed. Note that, as $n \in \mathbb{N}_0$, $d(t_1, t_2) \leq 2$ for all $t_1, t_2 \in T$. Also note that this number may be negative because of a large dependency depth n.

A compacted schedule (σ, μ) can be found by finding for all t the smallest values $p(t)$ that satisfy the derived constraints in (9). These values can be computed by solving the constraint graph (T, E) where edges E correspond to the task dependencies. (We assume without loss of generality that there is at most one task dependency in the TRM between a given source and destination. If multiple dependencies with different depths are present, then the one with the lowest depth is dominant.) An edge from t_1 to t_2 in (T, E) is labeled with weight $d(t_1, t_2)$. Following a variation of Theorem 24.9 in [13], we know the smallest value of $p(t)$ corresponds to the length of the longest path from any other task in T to t in graph (T, E). Such a solution exists if there are no positive-weight cycles in the graph. A key observation is that the existence of the feasible schedule σ' with period μ ensures that there are no positive cycles in this constraint graph (T, E). The weights represent the minimal separation in whole periods between identical repetitions of two tasks. A positive cycle would imply that both tasks can only execute after the other completed its execution, which results in infeasibility. As there exists a feasible schedule (σ', μ), we therefore know there are no positive weight cycles. Therefore, p exists and the computed p, σ, and μ with $\sigma(t) = p(t)\mu + \hat{\sigma}(t)$ form a compacted feasible periodic schedule (σ, μ) that is obtained by shifting start times σ' with integer multiples of μ.

The final step is to prove the desired bound. $p(t)$ is defined by the longest path to t in the graph (T, E). Assume that for task t the longest path is $t^1, \ldots, t^l$, for $t^1, \ldots, t^l \in T$, with $t^l = t$. This path has no cycles. Hence, $l \leq |T|$. We obtain a bound on the start time $\sigma(t)$ as follows:

$$\begin{aligned} \sigma(t) &= p(t)\mu + \hat{\sigma}(t) \\ &= \{\text{substitute the longest path from } t^1 \text{ to } t^l \text{ for } p(t)\} \\ &\quad \left(\sum_{m=1}^{l-1} d(t^m, t^{m+1}) \right) \mu + \hat{\sigma}(t) \\ &\leq \{\text{substitute } d \text{ using (10)}\} \end{aligned}$$

$$
\begin{aligned}
&\left(\sum_{m=1}^{l-1}\left\lceil\frac{\hat{\sigma}(t^m)-\hat{\sigma}(t^{m+1})+e_\alpha(t^m)}{\mu}\right\rceil\right)\mu+\hat{\sigma}(t)\\
&\leq \{\text{loosen bound by eliminating ceiling function}\}\\
&\left(\sum_{m=1}^{l-1}(1+\frac{\hat{\sigma}(t^m)-\hat{\sigma}(t^{m+1})+e_\alpha(t^m)}{\mu})\right)\mu+\hat{\sigma}(t)\\
&= \{\text{eliminate brackets}\}\\
&(l-1)\mu+\left(\sum_{m=1}^{l-1}\hat{\sigma}(t^m)-\hat{\sigma}(t^{m+1})+e_\alpha(t^m)\right)+\hat{\sigma}(t)\\
&= \{\text{resolve summation, leaving only terms } t^1,\ t^l\}\\
&(l-1)\mu+\hat{\sigma}(t)-\hat{\sigma}(t^l)+\hat{\sigma}(t)+\sum_{m=1}^{l-1}e_\alpha(t^m)\\
&< \{t^l=t,\ \hat{\sigma}(t)<\mu \text{ for any } t\in T\}\\
&(l-1)\mu+\mu+\sum_{m=1}^{l-1}e_\alpha(t^m)\\
&\leq \{l\leq|T|,\ (7)\}\\
&|T|\mu+|R|\mu\\
&=(|T|+|R|)\mu
\end{aligned}
$$

This concludes the proof. □

The following example shows the intuition for this proof.

Example 2. *Reconsider the task model of Figure 3 with the resource model, allocation, and task execution times as in Example 1. One feasible schedule (σ', μ) for this TRM, shown in Figure 5, is as follows: $\sigma'(x) = 0, \sigma'(y) = 12, \sigma'(z) = 28, \mu = 5$. Task phases $\hat{\sigma}$ are $\hat{\sigma}(x) = 0, \hat{\sigma}(y) = 2, \hat{\sigma}(z) = 3$.*

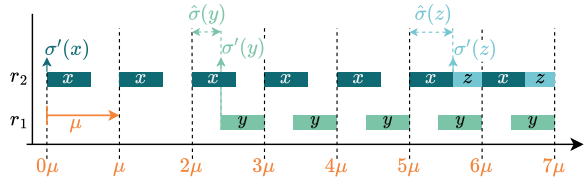

Fig. 5. Feasible schedule of the example TRM.

Observe that $\sigma'(x)$ falls in the first period $[0, \mu)$, $\sigma'(y)$ in the third period $[2\mu, 3\mu)$, and $\sigma'(z)$ only in $[5\mu, 6\mu)$ which is larger than our bound $(|T| + |R|)\mu = 5\mu$. The start time of the initial repetition of task x cannot be earlier as it already starts as early as possible. However, the initial repetition of task y can start directly after the initial repetition of task x and z can start after the initial repetition of y. Using (9), we can show that the compacted start time for the inital repetition of y can be derived as $\sigma(y) = p(y) \cdot \mu + \hat{\sigma}(y)$ with $p(y) = 1$ and for task z as $\sigma(z) = p(z) \cdot \mu + \hat{\sigma}(z)$ with $p(z) = 2$.

Consider the directed graph (T, E) with $T = \{x, y, z\}$ and E corresponding to the task dependencies with weights $d(t_1, t_2)$ from (10) for all $(t_1, n, t_2) \in D$, as shown in Table I. The resulting graph can be seen in Figure 6.

Start periods p are given by the longest paths from any task to the task at hand. In this case, for task x, the longest path starts from x itself. This path has length 0. For task y, the longest path is marked in orange in Figure 6. It is defined by $d(x, y)$ and has length 1. For task z, the longest path is found from x to z (marked in orange and teal) with length 2. We obtain that indeed $p(y) = 1$ and $p(z) = 2$. This gives us $\sigma(y) = p(y)\mu + \hat{\sigma}(y) = 7$, which is in the interval $[\mu, 2\mu)$, and $\sigma(z) = p(z)\mu + \hat{\sigma}(z) = 13$, which is in the interval $[2\mu, 3\mu)$. In line with Theorem 1, in the compact schedule, all start times are smaller than the bound $(|T|+|R|)\mu = 5\mu$. The compacted schedule (σ, μ) is the one already shown in Figure 4. ■

$(t_1, n, t_2) \in D$	$\hat{\sigma}(t_1) - \hat{\sigma}(t_2) + e_\alpha(t_1)$	$d(t_1, t_2)$
$(x, 0, y)$	0 - 2 + 3 = 1	$d(x, y) = 1 - 0 = 1$
$(y, 0, z)$	2 - 3 + 3 = 2	$d(y, z) = 1 - 0 = 1$
$(y, 3, x)$	2 - 0 + 3 = 5	$d(y, x) = 1 - 3 = -2$

TABLE I
CALCULATION OF THE WEIGHTS FOR TASK DEPENDENCIES

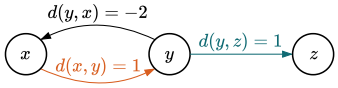

Fig. 6. Directed graph showing the periodic dependencies between tasks.

VI. Mixed-Integer Linear Programming Model

To solve the APSP, we can describe the problem as a monolithic Mixed-Integer Linear Programming (MILP) model. Using the bound from Theorem 1, it suffices to model a fragment of the optimal schedule where initial task repetitions start before $(|T| + |R|)\mu$. As each subsequent repetition is separated by μ time units, the fragment considers at most $(|T| + |R|)$ repetitions. The MILP model is as follows:

Parameters

(T, D, R, e)	TRM, as in Definition 1

Derived Parameters

$R_{t_1,t_2} = R_{t_1} \cap R_{t_2}$	The overlapping set of resources for two tasks $t_1, t_2 \in T$
$O = \{(t_1, t_2) \in T \mid t_1 \neq t_2, R_{t_1,t_2} \neq \emptyset\}$	Set of task pairs with an overlapping resource set
$\mathcal{B} = \lvert T\rvert + \lvert R\rvert$	Bound from Theorem 1
$\mathcal{I} = \mathbb{N}_0^{\leq \mathcal{B}}$	Domain for no-overlap constraint: task repetitions in the schedule fragment +1

Decision Variables

$\mu, s_t \in \mathbb{R}_0^+$	Period of the schedule and start time of task $t \in T$.

$$x_{t,r} = \begin{cases} 1 & \text{if } t \in T \text{ is allocated on resource } r \in R_t \\ 0 & \text{otherwise} \end{cases}$$

$$y_{t_1,t_2,k} = \begin{cases} 1 & \text{if repetition } k \in \mathcal{I} \text{ of task } t_1 \text{ is scheduled before the initial repetition of } t_2, \text{ where } (t_1, t_2) \in O \\ 0 & \text{otherwise} \end{cases}$$

The objective and constraints are defined as:

$$\min_{\mu,s,x,y} \quad \mu \tag{12}$$

$$\text{st} \quad \sum_{r \in R_t} x_{t,r} = 1, \qquad \forall t \in T, \tag{13}$$

$$\begin{aligned} s_{t_2} \geq s_{t_1} - n\mu + \sum_{r \in R_{t_1}} x_{t_1,r} \cdot e(t_1, r), \\ \forall (t_1,\, n,\, t_2) \in D, \end{aligned} \tag{14}$$

$$\begin{aligned} & s_{t_1} + e(t_1, r) + k\mu \\ \leq\ & s_{t_2} + M(3 - x_{t_1,r} - x_{t_2,r} - y_{t_1,t_2,k}), \\ & \forall (t_1, t_2) \in O, \forall r \in R_{t_1,t_2}, \forall k \in \mathcal{I}, \end{aligned} \tag{15}$$

$$\begin{aligned} & s_{t_2} + e(t_2, r) \\ \leq\ & s_{t_1} + k\mu + M(2 - x_{t_1,r} - x_{t_2,r} + y_{t_1,t_2,k}), \\ & \forall (t_1, t_2) \in O, \forall r \in R_{t_1,t_2}, \forall k \in \mathcal{I}, \end{aligned} \tag{16}$$

$$s_t + \sum_{r \in R_t} x_{t,r} \cdot e(t, r) \leq s_t + \mu, \qquad \forall t \in T, \tag{17}$$

$$s_t \leq \mathcal{B}\mu, \qquad \forall t \in T. \tag{18}$$

(12) specifies that the objective is to minimize the period of the schedule. (13) defines that there must be a resource allocation for each task as by Definition 2. (14) defines that tasks must adhere to the task dependencies as in (1).

(15) and (16) ensure a no-overlap constraint, as in (2), between every task on the same resource, across all repetitions within the domain $\mathcal{I}$. $\mathcal{I}$ covers one task repetition more than the repetitions being scheduled because task repetitions may complete after the time bound of Theorem 1. M is a sufficiently large integer, at least $\max_{t \in T, r \in R} e(t, r) \cdot |T| \cdot \mathcal{B}$. (15) ensures that the completion time of t_1 is at most the start time of t_2 if both t_1 and t_2 are scheduled on r and t_1 is scheduled before t_2. In that case, the multiplication with M results in 0; otherwise, it results in a large positive integer, trivially satisfying the constraint. (16) is a similar constraint covering the case that t_2 is scheduled before t_1. (17) specifies a no-overlap constraint between the firings of the same task. (18) ensures that all the first firings occur before the defined time bound.

Our model, unlike the models in [34, 33], offers a compact and efficient linear specification based on the bounded fragment of an optimal schedule. The models in [34, 33] capture a single-repetition schedule, which is a fragment of the schedule that considers a single repetition of each task. They require an additional set of decision variables that capture the relative distances, in whole periods, between the modeled task repetitions. These variables complicate the models by introducing non-linearity (multiplications between these distance variables and the period) and, after linearization, potential numerical instability. Our approach leads to a linear model by construction while preserving optimality.

VII. Benders Decomposition

The APSP is a complex combinatorial optimization problem. Monolithic models such as the model in Section VI result in a large number of constraints and variables. When scaling the size of the problem instance, even efficient solvers, such as IBM ILOG CPLEX Optimizer (CPLEX), tend to surpass acceptable time limits. A common strategy to improve performance involves decomposing the problem into smaller, easier-to-solve subproblems, by applying a *Benders decomposition* [1]. In addition to the monolithic model of Section VI, we present a model that applies a Benders decomposition. To speed up convergence of the iterative Benders decomposition, inspired by Quinton, Hamaz, and Houssin [33], we apply the acceleration techniques by Magnanti and Wong [25] and Papadakos [31].

A. Overview of the Approach

The general idea of the Benders decomposition is to separate the decision variables and their associated constraints of the *overall problem*, the APSP, into a *master* and *subproblem*. For the APSP, an MILP master problem MP determines a resource allocation and the ordering of all task repetitions on each resource. The corresponding Linear Programming (LP) subproblem SP then finds an optimal periodic schedule for the chosen allocation and ordering. Note that an LP problem can be solved efficiently, in contrast to an (M)ILP problem. Solving the master problem results in a set of trial values that are used to determine the next subproblem instance to solve. From the subproblem's solution, a Benders cut is generated—derived from the linear programming dual of the subproblem—and added to the master problem. These cuts refine the master problem by providing bounds on its objective, effectively guiding the search toward better solutions. This iterative process continues until the procedure has converged to the optimal solution to the overall problem. The mentioned acceleration techniques help select the most promising cuts.

B. Master Problem

We decompose the MILP model of Section VI. Master problem MP considers only the integer decision variables x and y from the model in Section VI, determining the allocation and ordering of tasks. A new decision variable $z \in \mathbb{R}_0^+$ is introduced that represents a lower bound on the optimal μ of the overall problem. The master problem minimizes z. The set of Benders cuts that are added during the iterative solving process is denoted $\mathcal{C}$, where each cut $c \in \mathcal{C}$ constrains the value of z. With this, we obtain a standard Benders decomposition, as formalized below with Eqs. (19)–(21).

However, such a standard Benders decomposition shows poor convergence, particularly during early iterations. Besides Benders cuts (21), the standard master problem only constrains the resource allocation x to a feasible one, constraint (20), and does not constrain the task ordering y. In such a standard Benders decomposition, the trial values $\bar{x}$ and $\bar{y}$ obtained by solving the master problem may often produce a resource allocation and task ordering that is not consistent with the

subproblem constraints, the task dependency and no-overlap constraints, resulting in an infeasible subproblem solution. The $\bar{y}$ trial values may for instance select multiple or conflicting orderings for a given pair of tasks. This slows down the convergence of the procedure towards an optimal solution of the overall problem.

Quinton, Hamaz, and Houssin [33] also observed these convergence issues in their Benders decomposition. To improve convergence, they add several predefined cuts to their master problem formulation. These predefined cuts ensure that the chosen resource allocation $\bar{x}$ and task ordering $\bar{y}$ are consistent for any feasible start times for a periodic schedule in the derived subproblem. The key idea is to enforce a task ordering $\bar{y}$ that is compatible with any feasible periodic schedule by deriving dummy start times for a periodic schedule with a sufficiently large period. This is achieved with predefined cuts, Eqs. (22)-(25) below, that are equivalent to the task dependency and no-overlap constraints of the overall problem, but using auxiliary decision variables w_t to represent dummy start times for every task $t \in T$ based on the constant upper bound $\hat{\mu}$ on the period defined as $\hat{\mu} = \sum_{t \in T} max_{r \in R_t}(e_{t,r})$ (i.e., the sum of the largest task execution times for al tasks).

We adopt the approach of [33], deriving predefined cuts reflecting the subproblem constraints that enforce a feasible resource allocation $\bar{x}$ and task ordering $\bar{y}$. This results in the following MP:

$$\min_{x,y,z,w} \quad z \tag{19}$$

$$\text{st} \quad \sum_{r \in R_t} x_{t,r} = 1, \qquad \forall t \in T, \tag{20}$$

$$z \geq c, \qquad \forall c \in \mathcal{C}, \tag{21}$$

$$w_{t_2} \geq w_{t_1} - n\hat{\mu} + \sum_{r \in R_{t_1}} x_{t_1,r} \cdot e_{t_1,r}, \quad \forall (t_1,\, n,\, t_2) \in D, \tag{22}$$

$$w_{t_1} + e_{t_1,r} + k\hat{\mu} \leq w_{t_2} + M(3 - x_{t_1,r} - x_{t_2,r} - y_{t_1,t_2,k}), \quad \forall (t_1,t_2) \in O, \forall r \in R_{t_1,t_2}, \forall k \in \mathcal{I}, \tag{23}$$

$$w_{t_2} + e_{t_2,r} \leq w_{t_1} + k\hat{\mu} + M(2 - x_{t_1,r} - x_{t_2,r} + y_{t_1,t_2,k}), \quad \forall (t_1,t_2) \in O, \forall r \in R_{t_1,t_2}, \forall k \in \mathcal{I}, \tag{24}$$

$$w_t + \sum_{r \in R_t} x_{t,r} \cdot e_{t,r} \leq w_t + \hat{\mu}, \qquad \forall t \in T. \tag{25}$$

Solving MP with decision variables x, y, z, w yields a solution $\bar{x}, \bar{y}, \bar{z}, \bar{w}$. MP can be solved more efficiently than the original overall problem because decision variable z does not occur in the constraints; the constraints use the constant $\hat{\mu}$ instead. The obtained allocation $\bar{x}$ and task ordering $\bar{y}$ are used in the next step of the procedure, solving a subproblem instance derived from this solution. The dummy start times $\bar{w}$ are discarded. The lower bound $\bar{z}$ is used to decide on convergence, as further explained later.

C. Subproblem

The LP subproblem SP in the Benders decomposition is obtained by fixing the integer variables x, y in the MILP from Section VI to the trial values $\bar{x}, \bar{y}$, i.e., fixing the resource allocation and ordering of task repetitions. SP is then:

$$\min_{\mu,s} \quad \mu \tag{26}$$

$$\text{st} \quad s_{t_2} \geq s_{t_1} - n\mu + \sum_{r \in R_{t_1}} \bar{x}_{t_1,r} \cdot e_{t_1,r}, \quad \forall (t_1,\, n,\, t_2) \in D, \tag{27}$$

$$s_{t_1} + e_{t_1,r} + k\mu \leq s_{t_2} + M(3 - \bar{x}_{t_1,r} - \bar{x}_{t_2,r} - \bar{y}_{t_1,t_2,k}), \quad \forall (t_1,t_2) \in O, \forall r \in R_{t_1,t_2}, \forall m \in \mathcal{I}, \tag{28}$$

$$s_{t_2} + e_{t_2,r} \leq s_{t_1} + k\mu + M(2 - \bar{x}_{t_1,r} - \bar{x}_{t_2,r} + \bar{y}_{t_1,t_2,k}), \quad \forall (t_1,t_2) \in O, \forall r \in R_{t_1,t_2}, \forall k \in \mathcal{I}, \tag{29}$$

$$s_t + \sum_{r \in R_t} \bar{x}_{t,r} \cdot e_{t,r} \leq s_t + \mu, \qquad \forall t \in T, \tag{30}$$

$$s_t \leq \mathcal{B}\mu, \qquad \forall t \in T. \tag{31}$$

We obtain the dual of SP, DSP, by introducing dual variables π^p, π^{nl}, π^{nr}, π^{nd}, π^s for contraints (27) up to (31), respectively. DSP is as follows. For compactness, the objective function is described by a function ob as defined in (35).

$$\max_{\substack{\pi^p,\pi^{nl},\pi^{nr},\\ \pi^{nd},\pi^s}} \quad ob(\pi^p, \pi^{nl}, \pi^{nr}, \pi^{nd}, \pi^s, \bar{x}, \bar{y}) \tag{32}$$

$$\begin{aligned} \text{st} \quad & \sum_{(t_1,n,t_2) \in D} n\pi^p_{t_1,n,t_2} \\ & - \sum_{\substack{(t_1,t_2) \in O,\\ r \in R_{t_1,t_2},\\ k \in \mathcal{I}}} k\pi^{nl}_{t_1,t_2,r,k} + k\pi^{nr}_{t_1,t_2,r,k} \\ & + \sum_{t \in T} \pi^{nd}_t + \mathcal{B}\pi^s_t \leq 1, \end{aligned} \tag{33}$$

$$\begin{aligned} & \sum_{(t_1,n,t) \in D} \pi^p_{t_1,n,t} - \sum_{(t,n,t_2) \in D} \pi^p_{t,n,t_2} \\ & + \sum_{\substack{(t,t_3) \in O,\\ r \in R_{t,t_3},\\ k \in \mathcal{I}}} \begin{pmatrix} \pi^{nl}_{t_3,t,r,k} - \pi^{nl}_{t,t_3,r,k} \\ + \pi^{nr}_{t,t_3,r,k} - \pi^{nr}_{t_3,t,r,k} \end{pmatrix} - \pi^s_t \leq 0, \\ & \qquad \forall t \in T. \end{aligned} \tag{34}$$

Solving DSP yields solution $\bar{\pi}^p$, $\bar{\pi}^{nl}$, $\bar{\pi}^{nr}$, $\bar{\pi}^{nd}$, $\bar{\pi}^s$ and objective $\overline{ob} = ob(\bar{\pi}^p, \bar{\pi}^{nl}, \bar{\pi}^{nr}, \bar{\pi}^{nd}, \bar{\pi}^s, \bar{x}, \bar{y})$. Upon solving DSP, if the procedure has not converged to an optimal

$$
\begin{aligned}
ob(\pi^p, \pi^{nl}, \pi^{nr}, \pi^{nd}, \pi^s, x, y) = & \sum_{(t_1,\, n,\, t_2) \in D} \pi^p_{t_1,n,t_2} \sum_{r \in R(t_1)} x_{t_1,r} \cdot e_{t_1,r} \\
& + \sum_{\substack{(t_1,t_2)\in O,\\ r \in R_{t_1,t_2},\\ k \in \mathcal{I}}} \pi^{nl}_{t_1,t_2,r,k} \left(e_{t_1,r} - M(3 - x_{t_1,r} - x_{t_2,r} - y_{t_1,t_2,k}) \right) \\
& + \sum_{\substack{(t_1,t_2)\in O,\\ r \in R_{t_1,t_2},\\ k \in \mathcal{I}}} \pi^{nr}_{t_1,t_2,r,k} \left(e_{t_2,r} - M(2 - x_{t_1,r} - x_{t_2,r} + y_{t_1,t_2,k}) \right) \\
& + \sum_{t \in T} \pi^{nd}_t \sum_{r \in R_t} x_{t,r} \cdot e_{t,r}
\end{aligned} \tag{35}
$$

solution, the traditional Benders decomposition adds a cut $ob(\bar{\pi}^p, \bar{\pi}^{nl}, \bar{\pi}^{nr}, \bar{\pi}^{nd}, \bar{\pi}^s, x, y)$ to the set of cuts $\mathcal{C}$, after which the procedure continues to solve the new master problem instance. Decision variables x and y remain open decision varaiables in a cut, making each cut a constraint on variable z in (21). However, SP is a degenerate subproblem, i.e., it does not have a unique solution $\bar{\pi}^p$, $\bar{\pi}^{nl}$, $\bar{\pi}^{nr}$, $\bar{\pi}^{nd}$, $\bar{\pi}^s$ that results in the objective $\overline{ob}$. Magnanti and Wong [25] determined that degenerate subproblems can lead to many different potential cuts. By picking the strongest cut, i.e., the one that most tightly constrains the master problem, convergence of the Benders decomposition can be significantly accelerated.

D. Accelerating the Benders Decomposition

According to Magnanti and Wong [25], the strongest cuts in a Benders decomposition with a degenerate subproblem lie on a so-called core point, a representative interior point of the dual solution space. The strongest cut is then found by solving an auxiliary (dual) subproblem, the Magnanti-Wong subproblem, that is obtained by substituting into the dual subproblem's objective function core points for the trial values and by constraining the objective value as computed in the dual subproblem to the already obtained objective value. Effectively, this recomputes a solution for the dual subproblem without changing its objective value.

We choose the same derivation for core points as [33], which applies the techniques of Papadakos [31], where the core point $\bar{v}^0$ for a variable v in iteration k of the Benders decomposition is defined as $\bar{v}^0_k = \frac{1}{2}\bar{v}^0_{k-1} + \frac{1}{2}\bar{v}_k$ for iterations $k > 0$ and $\bar{v}^0_0 = \frac{1}{2}\bar{v}_k$ for $k = 0$.

With this, in our case, the Magnanti-Wong subproblem ASP is obtained by computing core points $\bar{x}^0$, $\bar{y}^0$ from solutions $\bar{x}$ and $\bar{y}$ of the MP instance solved in the current iteration of the Benders decomposition and using the objective value $\overline{ob}$ from the dual subproblem DSP:

$$
\max_{\substack{\pi^p, \pi^{nl}, \pi^{nr},\\ \pi^{nd}, \pi^s}} \quad ob(\pi^p, \pi^{nl}, \pi^{nr}, \pi^{nd}, \pi^s, \bar{x}^0, \bar{y}^0) \tag{36}
$$

$$
\begin{aligned}
\text{st} \quad & ob(\pi^p, \pi^{nl}, \pi^{nr}, \pi^{nd}, \pi^s, \bar{x}, \bar{y}) = \overline{ob}, \\
& \text{constraint (33)}, \\
& \text{constraint (34)}
\end{aligned} \tag{37}
$$

Solving ASP renews the solution $\bar{\pi}^p$, $\bar{\pi}^{nl}$, $\bar{\pi}^{nr}$, $\bar{\pi}^{nd}$, $\bar{\pi}^s$. It results in the cut $ob(\bar{\pi}^p, \bar{\pi}^{nl}, \bar{\pi}^{nr}, \bar{\pi}^{nd}, \bar{\pi}^s, x, y)$.

The complete Benders decomposition procedure with Magnanti-Wong acceleration is shown in Algorithm 1. The algorithm terminates when the lower bound (LB) and upper bound (UB) converge to an identical value. The parameter ε represents the solution precision, needed to resolve numerical precision issues in representing integer and real numbers as floating point numbers.

Algorithm 1: Magnanti-Wong and Papadakos Accelerated Benders Decomposition Procedure

1: $\mathcal{P} \leftarrow \emptyset$, UB $\leftarrow \infty$
2: **MP:** Solve MP to obtain $\bar{x}, \bar{y}$ and objective $\bar{z}$.
3: **DSP:** Substitute $\bar{x}, \bar{y}$ in DSP to obtain $\bar{\pi}^p$, $\bar{\pi}^{nl}$, $\bar{\pi}^{nr}$, $\bar{\pi}^{nd}$, $\bar{\pi}^s$ and objective $\overline{ob}$.
4: **ASP:** Find the Papadakos core points $\bar{x}^0, \bar{y}^0$ and use $\overline{ob}$ to solve ASP and renew $\bar{\pi}^p$, $\bar{\pi}^{nl}$, $\bar{\pi}^{nr}$, $\bar{\pi}^{nd}$, $\bar{\pi}^s$.
   $\mathcal{C} \leftarrow \mathcal{C} \cup \{ob(\bar{\pi}^p, \bar{\pi}^{nl}, \bar{\pi}^{nr}, \bar{\pi}^{nd}, \bar{\pi}^s, x, y)\}$,
   LB $\leftarrow \bar{z}$, UB $\leftarrow \min\{\text{UB}, \overline{ob}\}$
5: **if** UB $-$ LB $> \varepsilon$ **then**
6: **goto MP**
7: **end if**
8: **return** $\bar{x}, \bar{y}$ and $\bar{s}, \bar{\mu}$ obtained by solving SP for $\bar{x}, \bar{y}$.

VIII. Experimental Evaluation

We performed various experiments to evaluate how our bounded-fragment-based modeling approach differs from the reference models from the literature. We created a benchmark of 120 TRMs and evaluated the performance of all the models on three metrics. The first metric is the total number of benchmark TRMs for which each technique found an optimal solution (*optimality count*). The second metric is the *accuracy* of the solutions found, as we observed that numerical instabilities may cause deficiencies. The third metric is the time that the solver needs for the models (*solve time*).

A. Evaluated Models

For the experimental evaluation, we compare the six different models that allow us to find optimal solutions to the APSP. The first two models are those presented in this paper.

1) *Bounded Monolithic*: The monolithic model based on bounding a fragement of the optimal periodic schedule presented in Section VI.
2) *Bounded Benders*: The model based on the Magnanti-Wong and Papadakos accelerated Benders decomposition presented in Section VII.

The third and fourth models are those presented in our primary references Quinton, Hamaz, and Houssin [34, 33]. To enforce non-overlapping task execution on given resources across all task repetitions, [33] introduces a variable K_{t_1,t_2} for each pair of tasks $t_1, t_2 \in T$. This variable K_{t_1,t_2} represents the relative distance, measured in whole periods, between identical repetitions of tasks t_1 and t_2. Similar to constraints (15) and (16) that require a multiplication of a constant $k \in \mathcal{I}$ with period μ, models in [33] require a multiplication of decision variables K_{t_1,t_2} and μ, resulting in non-linearity. To obtain a valid MILP, [33] linearizes the model by replacing μ by its inverse and solving for normalized start times u_t, equal to s_t/μ. For the experimental evaluation, we take the MILP presented in [33], which is a simplification of the one in [34], and a model applying the Magnanti-Wong and Papadakos accelerated Benders decomposition of [33].

3) *Quinton Monolithic*: The monolithic model of [33].
4) *Quinton Benders*: The Magnanti-Wong and Papadakos accelerated Benders decomposition of [33].

Using Theorem 1, every K_{t_1,t_2} variable can also be bounded as $-\mathcal{B} \leq K_{t_1,t_2} \leq \mathcal{B}$ without compromising optimality of the solution. This narrows the domain of K_{t_1,t_2} and therefore reduces the solution space compared to the original model in [33]. Applying our bound to both models in [33] results in the last two *hybrid* models.

5) *Bounded Quinton Monolithic*: The monolithic model of [33] with $-\mathcal{B} \leq K_{t_1,t_2} \leq \mathcal{B}$.
6) *Bounded Quinton Benders*: The Magnanti-Wong and Papadakos accelerated Benders decomposition of [33] with $-\mathcal{B} \leq K_{t_1,t_2} \leq \mathcal{B}$.

B. Experimental Setup

A benchmark of 120 TRM instances was generated using the SDF^3 [38] tool suite. The benchmark consists of two groups, each containing three sets of 20 unique graphs. The groups assume a fixed number of 10 and 15 tasks per graph, respectively. All task durations are randomly chosen between 1 up to 2000 time units. As in the experimental setup in [33], within each group, each set considers a decreasing number of 4, 3, and 2 resources.

All the models were implemented in Python and use the IBM ILOG CPLEX Optimizer (CPLEX) solver (22.1.1) to search for the optimal solutions. Solutions assume an optimality tolerance of 10^{-6}. For monolithic models, this is enforced through the solver settings, and for Benders decompositions, this is reflected in the value of ε in Algorithm 1. Subproblems of the Benders decomposition are solved with a tolerance of 10^{-9} to ensure that the ε bound in Algorithm 1 is sufficient for convergence. All experiments were conducted on an Intel i7-13700H 2.4GHz processor with 16 GB RAM. Each experiment was done with a 3600-second time limit. The respective benchmark instances, model code, and benchmark results can be found at https://github.com/TUE-EE-ES/APSP-toolset/releases/tag/1.0.2.

C. Experimental Results

We start by evaluating the optimality count of each technique, shown in Figure 7.

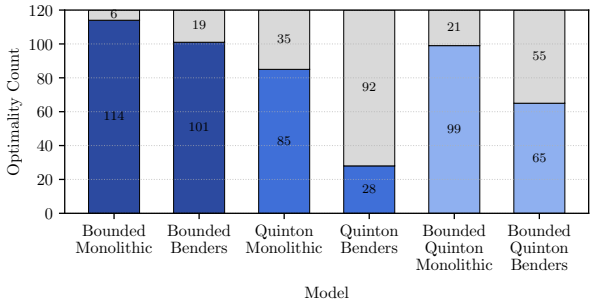

Fig. 7. Optimality count for each modeling technique.

The figure shows that our Bounded Monolithic model finds 114 optimal solutions out of the 120 cases, followed by our Bounded Benders model (101), and our bounded variant of the Quinton monolithic model (99). The Bounded Monolithic model finds 29 extra optimal solutions, 24 percentage points better, compared to the Quinton Monolithic model, the best performing reference model of [33]. This corresponds to an improvement of 34%. While decomposition-based models showed favorable performance in [33], for our benchmark, the monolithic models outperform their decomposition-based counterparts. Nevertheless, the Benders-decomposition-based models occasionally find optimal solutions for instances where the monolithic models reach the time limit.

Extra scalability experiments (generated using SDF^3) with 10 TRMs with 30 tasks on 8 and 16 resources and a 3600-second timeout on the Bounded Monolithic and Bounded Benders model confirm the superiority of the monolithic model. It finds optimal solutions in all cases, where the Bounded Benders model only finds eight, mainly due to a combination of time-consuming subproblems and insufficiently effective cuts. Improved cuts and/or acceleration techniques may improve the competitiveness of the Benders decomposition.

Returning to our original benchmark, when no optimal solution is found, this either means that the solver exceeded the time limit or that an *accuracy issue* occurred. Such accuracy issues may arise due to numerical stability problems and are identified through three different checks. First, the obtained solutions are validated to ensure that they satisfy all constraints (*constraint violation* issue). Second, for each model, the reported optimal objective values are compared to one another to identify deviating results (*deviating objective* issue). Third, models are compared based on whether they successfully complete the solving process without resulting in any solver issues or infeasibilities (*solving error* issue).

Figure 8 shows, for each benchmark set and model (marked on top as #tasks-#resources), the distribution of instances for which an optimal solution could be found, instances for which the solver reached the time limit, and instances that suffered from one of the three accuracy issues explained above. The figure shows that our models reported no accuracy issues

and reached the time limit for only a few problem instances, whereas all four other models suffer from accuracy issues.

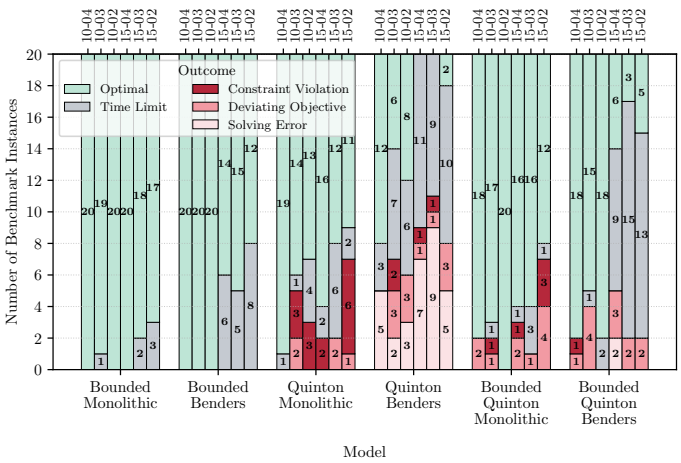

Fig. 8. Number of accuracy issues for each modeling technique.

Both the Quinton and bounded Quinton models resulted in solutions with constraint violations, incorrect claims of optimality, or errors in the solving process. These accuracy issues are hypothesized to stem from numerical instabilities. The use of floating-point arithmetic in MILP solvers like CPLEX poses a risk of rounding errors that may compromise the optimality guarantees in the solving process [14]. All of the issue-reporting models are based on linearizing a nonlinear problem formulation. Here, the linearized decision variables, which are normalized start times, can have small values requiring high decimal precision, while other decision variables or model inputs might be significantly larger. A minor rounding error in one of these small decision variables could escalate, leading to constraint violations, incorrect claims of optimality, and errors during the solving process.

In the monolithic models, we see that the numerical instabilities do not break the solving process but lead to solutions with constraint violations. In the Benders-decomposition-based models, numerical instabilities in one of the subproblems lead to infeasibilities in another, thus terminating the process prematurely and resulting in an error during the solving process.

Besides the optimality count and solution accuracy, we evaluate the respective solve times for each model. Figure 9 shows a boxplot of the solve times for each of the models. The boxes show the 20th to the 80th interpercentile range, and the whiskers show the 10th to the 90th interpercentile range. Only the solve times of 24 benchmark instances where every model has found an optimal solution are considered in the figure.

In terms of solve time, all the monolithic models outperform the Benders-decomposition-based models. However, as mentioned before, these models occasionally identify optimal solutions that are not obtained by their monolithic counterparts. So it may be worthwhile to invest in making these decompositions more effective. The solve times for the three monolithic models are all of the same order of magnitude. Given the limited data set of only 24 benchmark instances, no single model emerges as definitively superior in performance.

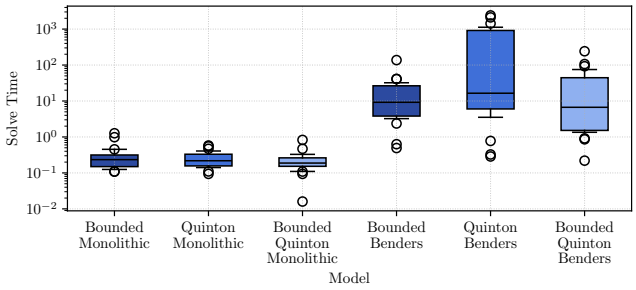

Fig. 9. Solve time for each modeling technique

We may conclude that all monolithic models demonstrate competitive solve times.

Overall, it may be concluded that the Bounded Monolithic and Bounded Benders models introduced in this paper outperform the state of the art in optimality count and numerical stability, while being competitive in solve time.

IX. Conclusion

In this work, we present models to find an optimal resource allocation and a strict-periodic schedule for a set of tasks with precedence constraints on heterogeneous resources. We proved that there exists a bound on the duration of a fragment of an optimal schedule in which all tasks execute at least once that does not impact solution optimality. It suffices to solve the scheduling problem within this bounded fragment, resulting in a compact linear constraint specification. Based on this specification, a monolithic MILP and a Benders decomposition using the Magnanti-Wong and Papadakos acceleration techniques are specified.

Experimental evaluation shows that our models outperform related state-of-the-art models of [33] in finding optimal solutions and numerical stability on a generated benchmark. Our models find optimal solutions for more benchmark instances compared to the models in [33]. Furthermore, our models do not experience numerical instabilities. The models of [33] are based on a non-linear specification that is linearized to obtain a valid MILP model. Our experiments show that these models may lead to various numerical instabilities in their solutions, causing constraint violations, incorrect claims of optimality, and errors in the solving process. Lastly, the experiments show that the models applying our bounding technique have competitive solve times.

This work can be extended in several ways. First, the scalability of the Benders decomposition can be improved. One approach is to find better accelerators or core-point selection methods to accelerate the convergence of the Benders procedure to an optimal solution. Additionally, implementing a Logic-Based Benders Decomposition [18] allows for the application-specific creation of Benders cuts, potentially further improving the convergence rate. Second, the problem itself can be refined to capture scenarios that cover constraints related to additional resources such as storage or memory resources or transportation or communication resources. This would require additional decision variables, thereby complicating the problem. Such extensions could potentially benefit from the suggested improved decomposition approaches.

Acknowledgement

This work received funding from the European Chips Joint Undertaking under Framework Partnership Agreement No 101139789 (HAL4SDV).

References

[1] J. F. Benders. "Partitioning procedures for solving mixed-variables programming problems". In: *Numerische Mathematik* 4.1 (Dec. 1962), pp. 238–252. ISSN: 0945-3245. DOI: 10.1007/BF01386316.

[2] Bruno Bodin and Alix Munier Kordon. "Evaluation of the exact throughput of a synchronous dataflow graph". In: *Journal of Signal Processing Systems* 93.9 (Sept. 2021), pp. 1007–1026. ISSN: 1939-8115. DOI: 10.1007/s11265-021-01649-z.

[3] Bruno Bodin, Alix Munier-Kordon, and Benoît Dupont de Dinechin. "K-periodic schedules for evaluating the maximum throughput of a synchronous dataflow graph". In: *Proc. 12th Int. Conf. on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*. July 2012, pp. 152–159. DOI: 10.1109/SAMOS.2012.6404169.

[4] Alessio Bonfietti. "Constraint based methods for allocation and scheduling of periodic applications". Doctoral Thesis. Alma Mater Studiorum - Università di Bologna, Apr. 19, 2013. DOI: 10.6092/unibo/amsdottorato/5503.

[5] Alessio Bonfietti et al. "A constraint based approach to cyclic RCPSP". In: *Proc. 17th Int. Conf. on Principles and Practice of Constraint Programming (CP)*. Springer, 2011, pp. 130–144. ISBN: 978-3-642-23786-7. DOI: 10.1007/978-3-642-23786-7_12.

[6] Alessio Bonfietti et al. "An efficient and complete approach for throughput-maximal SDF allocation and scheduling on multi-core platforms". In: *Proc. Design, Automation & Test in Europe (DATE)*. Mar. 2010, pp. 897–902. DOI: 10.1109/DATE.2010.5456924.

[7] Wojciech Bożejko, Jarosław Pempera, and Mieczysław Wodecki. "Parallel simulated annealing algorithm for cyclic flexible job shop scheduling problem". In: *Proc. Artificial Intelligence and Soft Computing (ICAISC)*. June 2015, pp. 603–612. ISBN: 978-3-319-19369-4. DOI: 10.1007/978-3-319-19369-4_53.

[8] Wojciech Bożejko et al. "Parallel tabu search for the cyclic job shop scheduling problem". In: *Computers & Industrial Engineering* 113 (Nov. 2017), pp. 512–524. ISSN: 0360-8352. DOI: 10.1016/j.cie.2017.09.042.

[9] Peter Brucker and Thomas Kampmeyer. "A general model for cyclic machine scheduling problems". In: *Discrete Applied Mathematics* 156.13 (July 2008), pp. 2561–2572. ISSN: 0166-218X. DOI: 10.1016/j.dam.2008.03.029.

[10] Peter Brucker and Thomas Kampmeyer. "Tabu search algorithms for cyclic machine scheduling problems". In: *Journal of Scheduling* 8.4 (July 2005), pp. 303–322. ISSN: 1099-1425. DOI: 10.1007/s10951-005-1639-4.

[11] Jinchao Chen et al. "Non-preemptive scheduling of periodic tasks with data dependencies in heterogeneous multiprocessor embedded systems". In: *ACM Transactions on Design Automation of Electronic Systems* 30.2 (Mar. 2025), pp. 1–25. ISSN: 1084-4309, 1557-7309. DOI: 10.1145/3711849.

[12] Jinchao Chen et al. "Work-in-progress: non-preemptive scheduling of periodic tasks with data dependency upon heterogeneous multiprocessor platforms". In: *Proc. 40th IEEE Real-Time Systems Symposium (RTSS)*. Dec. 2019, pp. 540–543. DOI: 10.1109/RTSS46320.2019.00059.

[13] Thomas H. Cormen et al. *Introduction to algorithms*. Third edition. Cambridge, Massachusetts London, England: MIT Press, 2009. ISBN: 978-0-262-03384-8 978-0-262-27083-0.

[14] Leon Eifler and Ambros Gleixner. "A computational status update for exact rational mixed integer programming". In: *Mathematical Programming* 197.2 (Feb. 2023), pp. 793–812. ISSN: 1436-4646. DOI: 10.1007/s10107-021-01749-5.

[15] Claire Hanen. "Study of a NP-hard cyclic scheduling problem: the recurrent job-shop". In: *European Journal of Operational Research* 72.1 (Jan. 1994), pp. 82–101. ISSN: 03772217. DOI: 10.1016/0377-2217(94)90332-8.

[16] Claire Hanen and Alix Munier. "A study of the cyclic scheduling problem on parallel processors". In: *Discrete Applied Mathematics* 57.2 (Feb. 1995), pp. 167–192. ISSN: 0166-218X. DOI: 10.1016/0166-218X(94)00102-J.

[17] Richard Hladík, Anna Minaeva, and Zdeněk Hanzálek. "On the complexity of a periodic scheduling problem with precedence relations". In: *Proc. Combinatorial Optimization and Applications (COCOA)*. Dec. 2020, pp. 107–124. ISBN: 978-3-030-64843-5. DOI: 10.1007/978-3-030-64843-5_8.

[18] John Hooker. *Logic-based Benders decomposition: theory and applications*. Cham: Springer International Publishing, 2024. ISBN: 978-3-031-45038-9 978-3-031-45039-6. DOI: 10.1007/978-3-031-45039-6.

[19] Jan Korst et al. "Periodic multiprocessor scheduling". In: *Proc. Parallel Architectures and Languages Europe (PARLE)*. June 1991, pp. 166–178. ISBN: 978-3-662-25209-3. DOI: 10.1007/978-3-662-25209-3_12.

[20] Alexandros Koulouris and Georgios P. Georgiadis. "An exact algorithm for calculating the minimum and feasible ranges of cycle time in periodic scheduling with shared resources". In: *Computers & Chemical Engineering* 175 (July 2023), p. 108286. ISSN: 00981354. DOI: 10.1016/j.compchemeng.2023.108286.

[21] E.A. Lee and D.G. Messerschmitt. "Synchronous data flow". In: *Proceedings of the IEEE* 75.9 (Sept. 1987), pp. 1235–1245. ISSN: 1558-2256. DOI: 10.1109/PROC.1987.13876.

[22] Eugene Levner et al. "Complexity of cyclic scheduling problems: A state-of-the-art survey". In: *Computers &*

Industrial Engineering 59.2 (Sept. 2010), pp. 352–361. ISSN: 0360-8352. DOI: 10.1016/j.cie.2010.03.013.

[23] Weichen Liu, Zonghua Gu, and Yaoyao Ye. "Efficient SAT-based application mapping and scheduling on multiprocessor systems for throughput maximization". In: *Proc. 18th Int. Conf. on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*. Oct. 2015, pp. 127–136. DOI: 10.1109/CASES.2015.7324553.

[24] Weichen Liu et al. "Efficient SAT-based mapping and scheduling of homogeneous synchronous dataflow graphs for throughput optimization". In: *Proc. 29th IEEE Real-Time Systems Symposium (RTSS)*. Nov. 2008, pp. 492–504. DOI: 10.1109/RTSS.2008.49.

[25] T. L. Magnanti and R. T. Wong. "Accelerating Benders decomposition: algorithmic enhancement and model selection criteria". In: *Operations Research* 29.3 (June 1981), pp. 464–484. ISSN: 0030-364X. DOI: 10.1287/opre.29.3.464.

[26] Ruben Mascaro et al. "Towards automating construction tasks: large-scale object mapping, segmentation, and manipulation". In: *Journal of Field Robotics* 38.5 (2021), pp. 684–699. ISSN: 1556-4967. DOI: 10.1002/rob.22007.

[27] Anna Minaeva and Zdeněk Hanzálek. "Survey on periodic scheduling for time-triggered hard real-time systems". In: *ACM Computing Surveys* 54.1 (Mar. 2021), 23:1–23:32. ISSN: 0360-0300. DOI: 10.1145/3431232.

[28] Orlando Moreira, Frederico Valente, and Marco Bekooij. "Scheduling multiple independent hard-real-time jobs on a heterogeneous multiprocessor". In: *Proc. 7th ACM & IEEE Int. Conf. on Embedded Software (EMSOFT)*. Sept. 2007, p. 57. ISBN: 978-1-59593-825-1. DOI: 10.1145/1289927.1289941.

[29] Martín Palos et al. "LiDAR-based perception system for logistics in industrial environments". In: *Applied Intelligence* 55.10 (Apr. 2025), p. 696. ISSN: 1573-7497. DOI: 10.1007/s10489-025-06528-9.

[30] Miloš Panić et al. "RunPar: an allocation algorithm for automotive applications exploiting runnable parallelism in multicores". In: *Proc. Int. Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*. Oct. 2014, pp. 1–10. ISBN: 978-1-4503-3051-0. DOI: 10.1145/2656075.2656096.

[31] Nikolaos Papadakos. "Practical enhancements to the Magnanti–Wong method". In: *Operations Research Letters* 36.4 (July 2008), pp. 444–449. ISSN: 0167-6377. DOI: 10.1016/j.orl.2008.01.005.

[32] K.K. Parhi and D.G. Messerschmitt. "Static rate-optimal scheduling of iterative data-flow programs via optimum unfolding". In: *IEEE Transactions on Computers* 40.2 (Feb. 1991), pp. 178–195. ISSN: 1557-9956. DOI: 10.1109/12.73588.

[33] Félix Quinton, Idir Hamaz, and Laurent Houssin. "A mixed integer linear programming modelling for the flexible cyclic jobshop problem". In: *Annals of Operations Research* 285.1 (Feb. 2020), pp. 335–352. ISSN: 1572-9338. DOI: 10.1007/s10479-019-03387-9.

[34] Félix Quinton, Idir Hamaz, and Laurent Houssin. "Algorithms for the flexible cyclic jobshop problem". In: *Proc. 14th IEEE Int. Conf. on Automation Science and Engineering (CASE)*. Aug. 2018, pp. 945–950. DOI: 10.1109/COASE.2018.8560449.

[35] Robin Roundy. "Cyclic schedules for job shops with identical jobs". In: *Mathematics of Operations Research* 17.4 (Nov. 1992), pp. 842–865. ISSN: 0364-765X, 1526-5471. DOI: 10.1287/moor.17.4.842.

[36] Paolo Serafini and Walter Ukovich. "A mathematical model for periodic scheduling problems". In: *SIAM Journal on Discrete Mathematics* 2.4 (Nov. 1989), pp. 550–581. ISSN: 0895-4801. DOI: 10.1137/0402049.

[37] Sundararajan Sriram and Shuvra S. Bhattacharyya. *Embedded multiprocessors: scheduling and synchronization*. 2nd ed. CRC Press, Oct. 2018. ISBN: 978-1-315-21975-2. DOI: 10.1201/9781420048025.

[38] S. Stuijk, M. Geilen, and T. Basten. "SDF^3: SDF for free". In: *Proc. 6th Int. Conf. on Application of Concurrency to System Design (ACSD)*. June 2006, pp. 276–278. DOI: 10.1109/ACSD.2006.23.

[39] S. Stuijk et al. "Multiprocessor resource allocation for throughput-constrained synchronous dataflow graphs". In: *Proc. 44th Design Automation Conference (DAC)*. June 2007, pp. 777–782. ISBN: 978-1-59593-627-1. DOI: 10.1145/1278480.1278674.