Grouping tasks to save energy in a cyclic scheduling problem: A complexity study

Claire C. Hanen, Zdenek Hanzalek

Grouping tasks to save energy in a cyclic scheduling problem: a complexity study

Claire Hanen

Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6, F-75005 Paris, France
UPL , Université Paris Nanterre , 92000, Nanterre, France

Zdenek Hanzalek

Czech Technical University in Prague
Czech Institute of Informatics, Robotics and Cybernetics
CZ-160 00 Prague 6, Czech Republic

Abstract

This paper is motivated by the repetitive and periodic transmission of messages in Wireless Sensor Networks (WSNs) given by the ZigBee standard. In order to save energy, communication tasks are grouped in time. The WSN applications, such as control loops used in production lines, impose deadlines on the message delivery time.

By introducing a grouping constraint, this paper extends the polynomial cyclic scheduling problem of building a periodic schedule with uniform precedence constraints and no resource constraints. We consider that each task belongs to one group, and each group is to be scheduled as a super-task in every period. We show two examples issued from different applications of such grouping constraints, using uniform constraints to model the deadlines expressed in the number of periods. We tackle the feasibility problem (the existence of a periodic schedule), which has shown to be independent of the processing times. We propose a graph formulation of the problem using the retiming technique.

In the ZigBee case, we prove that the particular tree structure of the groups and of the uniform precedences based upon the communication tree, lead to a polynomial algorithm to solve the problem. But the general problem is proven to be NP-complete even if no additional resource constraints are considered, and unit processing times are assumed. This extension of the cyclic scheduling leads to a new range of grouping problems with applications, not only in communication networks but also in production and logistics scheduling.

Keywords: scheduling, complexity theory, cyclic scheduling, task grouping, energy savings

1. Introduction

This paper was initially motivated by the periodic scheduling of messages in Wireless Sensor Networks (WSNs). Applications (such as industrial control systems with feedback loops) using WSNs define a set of messages to be periodically transmitted through the sensor network, along predefined routes eventually in opposite directions. The applications usually impose deadline constraints on a message delivery time

(called the start-to-end deadline in this paper), which makes the scheduling of communications interesting from an OR perspective, and relates it to cyclic scheduling problems.

The wireless devices are usually battery-powered, so the communications performed by the device during each period are grouped in time, in order to save energy. Thus, the main motivation of the paper was to model the grouping and to study how it affects the complexity of the underlying cyclic scheduling problem. The grouping may be considered not only in communication networks, but also in production and logistics (see the example in Subsection 2.3), so that the question we tackle in this paper is a general issue for cyclic scheduling.

Cyclic scheduling addresses problems where a set of tasks $\mathcal{T}$ has to be repeated iteratively and several repetitions may interleave in time. Such problems can be found in the computation loops of Dupont de Dinechin & Munier-Kordon (2014); Benabid & Hanen (2011); Hanzalek (1998) or in the embedded systems of Marchetti & Munier-Kordon (2009); Khatib et al. (2016); Bodin et al. (2016). Cyclic scheduling was successfully applied in robotic cells or the shop problems of Proth & Xie (1995); Levner et al. (2007); Alcaide et al. (2007); Gultekin et al. (2006); Bonfietti et al. (2011); Kats & Levner (2011b); Bocewicz et al. (2017); Kats & Levner (2018); Pempera & Smutnicki (2018). A cyclic variant of a hoist scheduling problem was studied in Che et al. (2015); Feng et al. (2018); Gao & Zhou (2019).

Most papers use a model with cyclic precedence constraints, due to the data flows or production process dependencies. The simplest and widely accepted model is the uniform precedence model of Munier (1996); Brucker & Kampmeyer (2005); Hanen (2009), which is used in this paper too.

One repetition of a set of tasks is called an iteration. A schedule assigns a starting time $s_{i,k}$ to each iteration $k \geq 1$ of each task i. So for each i, $(s_{i,k})_{k\geq 1}$ is an infinite increasing sequence. The usual criterion in cyclic scheduling is to minimize the average cycle time, i.e., the mean time interval between two iterations of the task set $\mathcal{T}$ $\lim\limits_{k\to\ +\infty} \dfrac{\max_{i\in\mathcal{T}} s_{i,k}}{k}$.

The concept of uniform precedences is used to control the interleaving of the iterations. Assuming uniform precedence constraints and no additional resource constraint, periodic schedules (for which an occurrence of task i starts every λ time units, i.e., $s_{ik} = s_i + (k-1)\lambda$) are dominating schedules (i.e., among the optimal schedules, at least one is periodic) with respect to the usual criteria of the average cycle time minimization. λ is then the average cycle time of the schedule. It is also called the period of the periodic schedule. Moreover, under these assumptions, there are polynomial algorithms to compute the minimum period λ of a periodic schedule in Munier (1996); Levner & Kats (1998); Hanen (2009).

Even if periodic schedules are no longer dominating in the presence of additional resource constraints, many authors consider the periodic schedule, since they are easy to implement and since approximation algorithms have been found for these problems. Among them we note the retiming based algorithms of Gasperoni & Schwiegelshohn (1994); Calland et al. (1998) that inspired the techniques developed in this paper. From the complexity point of view, a cyclic scheduling problem that combines resource constraints and arbitrary uniform precedence constraints is usually NP-hard, Hanen (1994), but some polynomial

special cases have been published, Hanen (2009); Kats & Levner (2011a).

In Hanzalek & Jurcik (2010) the scheduling problem induced by the WSN following the IEEE 802.15.4/ZigBee standard was studied, and it was observed that the problem could be formulated as a periodic cyclic scheduling problem, and that precedences induced by the data flow could be formulated as uniform precedence constraints. When considering multiple collision domains (that may be modeled as dedicated resources with multi-resource tasks) and exact start-to-end deadlines, we modeled and solved the problem by ILP (Integer Linear Programming).

Then in Hanzalek & Hanen (2015), "core precedence constraints" were introduced to model the general precedence relations between the tasks during an interval of length λ, the period, regardless of their iteration number. That was the first attempt to model one of the characteristics of the grouping of tasks. It was shown that the existence of a periodic schedule with uniform precedence constraints and core precedence constraints is an NP-hard problem, even without additional resource constraints. However, this model did not provide a polynomial solution to the ZigBee case.

In this paper, we propose a new model for the grouping in a cyclic context that is suitable for WSN, and was partly inspired by batching, or clustering.

Each task belongs to a group, as in Potts & Kovalyov (2000) where each task belongs to a family. However, unlike in Potts et al.'s paper, no setup times are considered to model the incentive to the group tasks of the same family into a batch. The grouping constraint imposes that at each period interval, all the tasks of the same group have to be processed together as a super-task, but the iteration number of the tasks composing the super-task is still to be determined. A precedence constraint between the two tasks belonging to the different groups is then applied to the corresponding super-tasks.

In Section 2, we address the general UGF (Uniform Grouped Feasibility) problem of the feasibility of a system combining uniform precedence constraints and grouping constraints. We describe a production example, and we use cyclic scheduling theory to express the UGF as a graph problem that does not depend on the processing times.

In Section 3, we address the WSN ZigBee standard as a special case of the UGF. We propose a model of the start-to-end deadlines expressed in a number of periods that allows them to be formulated as usual uniform precedence constraints. We then show that, due to the special structure of the groups, the existence of a feasible periodic schedule can be checked in polynomial time.

In Section 4, we show that the general UGF problem, defined in Section 2, is NP-complete, even if unit processing times are assumed.

2. General problem setting

In this section, we introduce the notations, we show an example from the area of production scheduling, and we define the features of the cyclic grouping problem addressed in the paper.

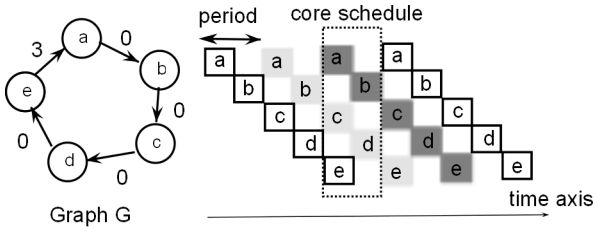
2.1. Uniform precedence model

We consider uniform precedence constraints, which are often used to model cyclic scheduling problems Hanen (2009). Furthermore, we extend it by introducing a new "grouping constraint".

A set of tasks $\mathcal{T}$ is given. The processing time of task i is denoted by p_i. The tasks are assumed to be infinitely repeated with no reentrance: the $(k+1)^{th}$ iteration of task i cannot start before the end of the k^{th} iteration of task i.

Definition 1. *A valued graph $G = (\mathcal{T}, A, L, H)$ of* **uniform precedence constraints** *is given as follows: an arc $a = (i, j)$ of A is characterized by two non-negative values called length $L(a) = l_{ij}$ and height $H(a) = h_{ij}$ of the arc. Arc a represents an infinite number of precedence constraints: for any integer k, the $(k + h_{ij}) - th$ iteration of task j cannot start earlier than l_{ij} times units after the beginning of the $k - th$ iteration of task i.*

The non-reentrance of i can be modeled by a uniform constraint $a = (i, i)$ with $L(a) = p_i$ and $H(a) = 1$. Below, these non-reentrance constraints remain implicit. Notice that unlike the noncyclic scheduling problem, G may contain cycles. For any path μ of G we denote by $L(\mu)$(resp. $H(\mu)$) the sum of the lengths (resp. heights) of the arcs of μ.



	a	b	c	d	e
s_i	0	1	2	3	4
$\sigma(i)$	0	1	0	1	0
$\alpha(i)$	0	0	1	1	2

Figure 1: A uniform graph G with unit lengths of the arcs and unit processing times, labeled with the heights of the arcs, and a periodic schedule of period $\lambda = 2$ with four iterations of the tasks

Definition 2. *A* **periodic schedule of period** λ *assigns a starting time s_i to the first iteration of each task i so that the starting time $s_{i,k}$ of the k-th iteration of i is :*

$$s_{ik} = s_i + (k-1)\lambda$$

Notice that s_i may be greater than λ, so that the completion time of the first iteration of tasks (i.e., $max_{\{i\in\mathcal{T}\}} s_i + p_i$) may be greater than several periods.

The uniform precedence constraints induced by G can then be expressed with the following inequality:

$$\forall (i,j) \in G,\ s_j - s_i \geq l_{ij} - \lambda h_{ij} \tag{1}$$

Let us define $\mathcal{C}(G)$ as a set of cycles in graph G. The necessary condition of the existence of a feasible schedule is that for any $C \in \mathcal{C}(G)$, $H(C) > 0$. Moreover, in any periodic schedule, $\lambda \geq \max_{C\in\mathcal{C}(G)} \frac{L(C)}{H(C)}$ Hanen (2009).

Definition 3. *A periodic schedule can also be described by its* **core** σ *and its* **retiming** α *as follows:* $\alpha(i)$ *is the non-negative integer quotient and* $\sigma(i)$ *is the non-negative integer remainder after dividing* s_i *by* λ *so that*

$$s_i = \sigma(i) + \lambda\alpha(i), \text{ with } \sigma(i) \in [0, \lambda). \tag{2}$$

In the rest of the paper we shall denote a schedule by a triple $(\sigma, \alpha, \lambda)$.

Let us consider an interval $[(x-1)\lambda, x\lambda]$ with a sufficiently large non-negative integer x. We call it the x^{th} execution period. Notice that at time $(x-1)\lambda + \sigma(i)$, the iteration $x - \alpha(i)$ of task i starts. Indeed, the starting time of this iteration of task i is $\sigma(i) + \lambda\alpha(i) + \lambda(x - \alpha(i) - 1) = (x-1)\lambda + \sigma(i)$.

For task i, the retiming $\alpha(i)$ defines the iteration number (i.e., $x - \alpha(i)$) of task i performed in the x^{th} execution period, whereas core σ defines the starting time of task i in any execution period regardless of the task iteration number.

Now consider an arc (i, j) of G, the corresponding uniform constraint is expressed with the following relation:

$$\sigma(j) - \sigma(i) \geq l_{ij} - \lambda(h_{ij} + \alpha(j) - \alpha(i)) \tag{3}$$

2.2. Grouping

We consider here the definition of what we call a grouping constraint, which defines the way the tasks have to be grouped as super-tasks.

Definition 4. *A* **grouping constraint** $\mathcal{Y}$ *is composed by*

- *an integer* K *representing the number of groups;*
- *an integer* $k_i \in 1, \ldots, K$ *representing the group label for each task* $i \in \mathcal{T}$*;*
- *an assumption that the subgraph of* G*, in which the nodes are the tasks of a given group* k*, is acyclic;*
- *a starting time* $g(i)$ *relative to the starting time of the group* k_i *(see below) for each task* $i \in \mathcal{T}$*;*
- *a processing time* π_k *of super-task* γ_k *associated to each group* $k \in 1, \ldots, K$*.*

This grouping constraint will induce constraints on the periodic schedule of the super-tasks.

Definition 5. *A* **grouped periodic schedule** *satisfying a grouping constraint* $\mathcal{Y}$ *is a periodic schedule such that the tasks of the same group are executed as a single super-task in the core schedule: for any task* $i \in \mathcal{T}$ *for which* $k_i = k$*, we can define* $\sigma(i) = \sigma(\gamma_k) + g(i)$ *(i.e. the starting time of* i *can be given relatively to the starting time of its super-task). Moreover, for each arc* (i, j) *of* G *such that* $k_i \neq k_j$*, the inequality (4), defined below, is met.*

The values of $g(i)$ for all $k_i = k$ are assumed to be compatible with the precedence constraints issued from the subgraph of G restricted to group k. In the model addressed in Section 3, the tasks of the same group are independent. Hence, in this case, we will assume that all the tasks composing an iteration of the super-task γ_k start at the same time in the core. But, in other applications, π_k could also be the sum of the processing times, or the maximum processing time (if the tasks of the same group are independent).

If we consider the x^{th} period interval, for a sufficiently large x, the iteration numbers of the tasks composing the super-task performed in this interval might be different from each other. So the retimings of the tasks composing a group may have different values.

We now define inequality (4) that highlights the relationship between a grouped periodic schedule and the uniform graph G. Let $(\sigma, \alpha, \lambda)$ be a grouped periodic schedule.

Consider a uniform precedence constraint (i, j) with $k_i \neq k_j$ then the iteration of the super-task γ_{k_i} embedding the iteration q of i precedes (with length $l_{ij} + \pi_{k_i} - p_i$) the iteration of the super-task γ_{k_j} embedding the iteration $q + h_{ij}$ of j so that it induces a constraint between the super-tasks :

$$\sigma(\gamma_{k_j}) + \lambda\alpha(j) - \sigma(\gamma_{k_i}) - \lambda(\alpha(i)) \geq l_{ij} + \pi_{k_i} - p_i - \lambda h_{ij} \tag{4}$$

As the position of the task within a super-task is not known, inequality (4) considers the worst positions, i.e. the completion of task i at the completion time of super-task γ_{k_i}, and the start of task j at the start time of super-task γ_{k_j}.

We address the following question: Given a uniform precedence graph G and a grouping constraint $\mathcal{Y}$ does a grouped periodic schedule of G exist? We call this decision problem the UGF (Uniform Grouped Feasibility).

2.3. A production example

Let us consider the following example illustrated by Figure 2. There are three manufacturing plants I, II, III and three carriers 1, 2, 3 so that carrier 1 handles the grouped tasks from I to II, carrier 2 from II to III, and finally carrier 3 from III to I.

There are three (work) flows: the first flow, starting in plant I with manufacturing task a_1, is further transported (task b_1) by carrier 1 to plant II, where another manufacturing task c_1 is performed. Then it is transported (task d_1) by carrier 2 to plant III, where it is finished by the manufacturing task e_1. Similarly, the second flow starts with task a_2 in II, and it is transported by carrier 2 to III and by carrier 3 to I, where it is finished by task e_2. Finally, the third flow starts with task a_3 in III, and it is transported by carrier 3 to I and by carrier 1 to II, where it is finished by task e_3. The duration of each task is one hour.

The transportation tasks performed by a given carrier are grouped together and performed just once per period (the period is equal to one shift of 8 hours) to save energy. Each flow is executed periodically every shift and is constrained by the start-to-end deadline expressed in the number of traversed periods.

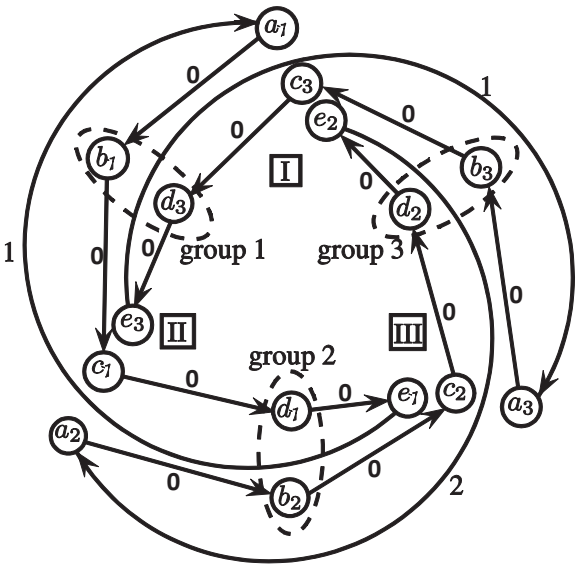


Figure 2: Graph G of an example with three manufacturing plants and three transport services (G has the unit lengths of the arcs, the unit processing times and labels standing for the heights of the arcs).

The start-to-end deadline has a value of one in the case of flows 1 and 3, it means the iteration of the flows has to start and finish within 8 hours. This implies that the k^{th} iteration of e_1 precedes the $k+1^{th}$ iteration of a_1, this results in a uniform precedence between e_1 and a_1 with height 1. Two shifts in the case of flow 2 means the flow has to start and finish within 16 hours.

The question is: how to schedule the operations and particularly the carriers so that the periodic production is feasible?

A feasible periodic schedule is represented in Figure 3. Please notice that the second flow spans two periods. If the second flow would have a start-to-end deadline of 1 shift, then no feasible grouped periodic schedule could be found, whatever period length (mid-day, day, month) or processing time (minute, second) is considered.

The schedule depicted in Figure 3 is defined by the following values:

retiming α	a_i	b_i	c_i	d_i	e_i
flow 1	0	0	0	0	0
flow 2	0	0	0	1	1
flow 3	0	0	0	0	0

core σ	a_i	b_i	c_i	d_i	e_i
flow 1	0	3	4	5	6
flow 2	0	5	6	1	2
flow 3	0	1	2	3	4

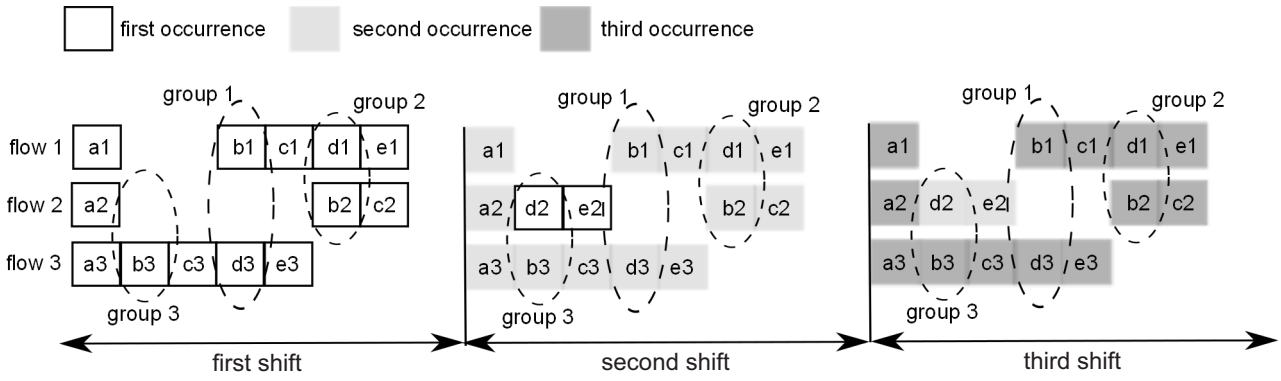


Figure 3: A periodic schedule of the example in Figure 2

2.4. A graph expression of UGF

According to Relation (3), we can recall important properties of the core of a periodic schedule satisfying the uniform precedence constraints, Darte & Huard (2000); Hanen (2009). Let s be a periodic schedule with retiming α and core σ.

Property 1. *For any arc (i,j) of G, $h_{ij} + \alpha(j) - \alpha(i) \geq 0$. Moreover, if $h_{ij} + \alpha(j) - \alpha(i) = 0$ then $\sigma(j) \geq l_{ij} + \sigma(i)$.*

Definition 6. *Following Calland et al. (1998), we say that the retiming α is* **a legal retiming***, if for any arc (i,j) of G, $h_{ij} + \alpha(j) - \alpha(i) \geq 0$.*

Property 2. *For any cycle C of G with height $H(C)$ there are, at most, $H(C)$ arcs for which $h_{ij} + \alpha(j) - \alpha(i) > 0$*

For a given legal retiming α, we define the retiming graph $\mathcal{R}^\alpha$ to be the sub-graph of G composed with all the arcs (i,j), so that $h_{ij} + \alpha(j) - \alpha(i) = 0$. The arcs are valued by l_{ij}. Notice that this retiming graph might have no arc. According to Relation (3), $\mathcal{R}^\alpha$ is an acyclic graph: a cycle $\mu = (i_1, \ldots, i_k, i_1)$ would induce a contradiction $\sigma(i_1) < \sigma(i_2) < \ldots < \sigma(i_k) < \sigma(i_1)$. The core σ meets the non cyclic precedence constraints defined by $\mathcal{R}^\alpha$. Indeed, if (i,j) is an arc of $\mathcal{R}^\alpha$, then

$$\sigma(j) - \sigma(i) \geq l_{ij} - \lambda(h_{ij} + \alpha(j) - \alpha(i)) = l_{ij}.$$

It has been shown (see Hanen (2009)) that for a given legal retiming α, according to Definition 6, and for any schedule σ meeting the precedence constraints of $\mathcal{R}^\alpha$, a value $\underline{\lambda}$ exists that can be computed in polynomial time so that for any $\lambda \geq \underline{\lambda}$ $(\sigma, \alpha, \lambda)$ there is a feasible periodic schedule.

We now have to express that the tasks of the same group are executed as a super-task in the core schedule:

Definition 7. *Let α be a given retiming, and $\mathcal{G}^\alpha$ be the graph (called the* **grouped retiming graph***) obtained from $\mathcal{R}^\alpha$ by merging the nodes of the same group.*

According to Relation (4), which defines the grouping constraint, for any arc (i,j) of $\mathcal{R}^\alpha$, with $k_i \neq k_j$, which results in an arc of $\mathcal{G}^\alpha$,

$$\sigma(\gamma_{k_j}) - \sigma(\gamma_{k_i}) \geq l_{ij} + \pi_{k_i} - p_i - \lambda(h_{ij} + \alpha(j) - \alpha(i)) = l_{ij} + \pi_{k_i} - p_i > 0$$

So an arc (i,j) of $\mathcal{R}^\alpha$ results in an arc $(\gamma_{k_i}, \gamma_{k_j})$ of $\mathcal{G}^\alpha$ with the value $l_{ij} + \pi_{k_i} - p_i$.

The feasibility problem can be expressed as follows:

Theorem 1. *An instance of UGF is feasible, if and only if, there is a legal retiming vector α so that the resulting grouped retiming graph $\mathcal{G}^\alpha$ is acyclic.*

Proof. Even if $\mathcal{R}^\alpha$ is acyclic, by merging the nodes, $\mathcal{G}^\alpha$ may contain cycles. So, in such a case, no feasible core schedule of the super-tasks can be built, since the core schedule of the super-tasks should respect the precedence relations defined by $\mathcal{G}^\alpha$. As observed above, for any arc (i,j) of $\mathcal{R}^\alpha$, with $k_i \neq k_j$, we should have

$$\sigma(\gamma_{k_j}) - \sigma(\gamma_{k_i}) \geq l_{ij} + \pi_{k_i} - p_i - \lambda(h_{ij} + \alpha(j) - \alpha(i)) = l_{ij} + \pi_{k_i} - p_i > 0$$

Conversely, suppose that $\mathcal{G}^\alpha$ is acyclic and consider a schedule σ of the super-tasks that satisfies the precedence relations induced by $\mathcal{G}^\alpha$. We can provide a core schedule of the original tasks by setting, for each task i, $\sigma(i) = \sigma(k_i) + g(i)$. For any arc (i,j) of G, several cases may occur:

- $k_i = k_j$, and by assumption $g(j) \geq g(i) + l_{ij}$ so that $g(j) - g(i) = \sigma(j) - \sigma(i) \geq l_{ij} \geq l_{ij} - \lambda(h_{ij} + \alpha(j) - \alpha(i))$ for any $\lambda \geq 0$, and, thus, the uniform constraint is satisfied for any period value.

- $k_j \neq k_i$ and (i,j) is an arc of $\mathcal{R}^\alpha$, i.e., $h_{ij} + \alpha(j) - \alpha(i) = 0$. We know that $\sigma(j) - \sigma(i) = \sigma(k_j) + g(j) - \sigma(k_i) - g(i)$. But $g(j) - g(i) \geq \pi_{k_i} - p_i$, since the processing time of the group is greater than the processing time of any task composing the group. Hence, $\sigma(j) - \sigma(i) \geq \sigma(k_j) - \sigma(k_i) - (\pi_{k_i} - p_i) \geq l_{ij}$ according to Relation (4) of the grouping constraint. No matter where the position of the tasks in the super tasks is, the precedence relation is satisfied. As $h_{ij} + \alpha(j) - \alpha(i) = 0$, the uniform constraint (i,j) is satisfied since Relation (3) is satisfied for any value of $\lambda \geq \max_{i \in \mathcal{T}} \sigma(i) + p_i$.

- Otherwise, $k_j \neq k_i$ and $h_{ij} + \alpha(j) - \alpha(i) \geq 1$. According to Relation (4) on the grouping constraint, we should have $\sigma(k_j) - \sigma(k_i) \geq l_{ij} + \pi_{k_i} - p_i - \lambda(h_{ij} + \alpha(j) - \alpha(i))$. This is always true assuming $\lambda \geq \sigma(k_i) - \sigma(k_j) + \pi_{k_i} - p_i + l_{ij}$. Moreover, for such a value of λ,

  $$\sigma(j) - \sigma(i) = \sigma(k_j) - \sigma(k_i) + g(j) - g(i) \geq \sigma(k_j) - \sigma(k_i) - (\pi_{k_i} - p_i) \geq l_{ij} - \lambda(h_{ij} + \alpha(j) - \alpha(i)).$$

  Hence, the original uniform constraint is satisfied.

Hence, there will be a value $\underline{\lambda}$ so that a grouped periodic schedule exists for any $\lambda \geq \underline{\lambda}$. ■

Notice that Theorem 1 shows that solving the UGF decision problem can be undertaken without considering the processing times of the tasks and super-tasks, since checking the existence of cycles in $\mathcal{G}^\alpha$ does not depend on the processing times. This is the reason why the unit processing times and unit

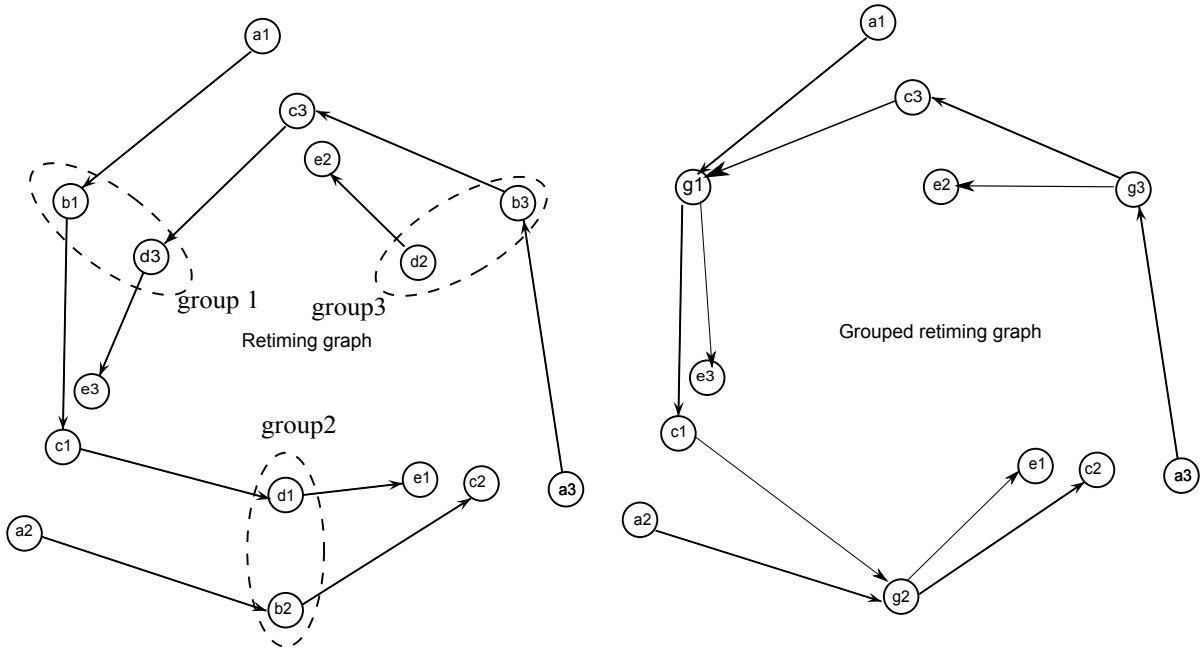


Figure 4: The retiming graph $\mathcal{R}^\alpha$ (left) and the grouped retiming graph $\mathcal{G}^\alpha$ (right)

length are assumed in the applications we consider in the paper (Section 2 and 3). However, of course, the processing times are to be considered when constructing a solution as in the Proof of Theorem 1.

Figure 4 shows graphs $\mathcal{R}^\alpha$ and $\mathcal{G}^\alpha$ for the periodic schedule depicted in Figure 3.

2.5. Basic properties of the UGF

The graph formulation of the UGF has some algorithmic outcomes on the properties of the grouped schedules

Property 3. *For a given legal retiming α, checking the existence of a periodic schedule $(\sigma, \alpha, \lambda)$ satisfying the grouping constraints can be undertaken in polynomial time.*

Proof.

Assume that we are given a retiming α so that for any arc (i, j) of G:

$$h_{ij} + \alpha_j - \alpha_i \geq 0$$

From α, we can build $\mathcal{R}^\alpha$, and, thus, the grouped retiming graph $\mathcal{G}^\alpha$. To know if a feasible grouped periodic schedule can be built from α, we just have to check whether $\mathcal{G}^\alpha$ is acyclic, which can be calculated by a depth first search in linear time Cormen et al. (2009). ■

Property 4. *If a core schedule σ of the super-tasks is given, checking the existence of a retiming α and a period λ so that $(\sigma, \alpha, \lambda)$ is a feasible periodic schedule can be undertaken in polynomial time.*

Proof. Once the schedule of the groups, and, hence, the core schedule σ, is known, notice that for any task i, $\lambda > \sigma(i)$ (otherwise σ could not be the core of the schedule). The only thing to check is the existence of a legal retiming α and a period λ satisfying Relation (3) for an arc $a = (i, j)$ of G :

$$\alpha_j - \alpha_i \geq \left\lceil \frac{l_{ij} + \sigma(i) - \sigma(j)}{\lambda} \right\rceil - h_{ij} \tag{5}$$

Let us denote $v_\lambda(a) = \left\lceil \frac{l_{ij}+\sigma(i)-\sigma(j)}{\lambda} \right\rceil - h_{ij}$

For a fixed value of λ, Relation 5 induces a difference constraint system, Cormen et al. (2009), i.e., a system of inequalities of the form $x - y \leq b$ (or in our case $x - y \geq b$ - which can also be expressed as $y - x \leq -b$ of the first form) with variables x, y and constant b which may be positive or negative.

It is known that a solution exists if and only if graph G has no cycles μ with $v_\lambda(\mu) > 0$. This can be checked in polynomial time by using the Bellman-Ford Algorithm presented in Cormen et al. (2009) inspired by the Bellman algorithm, Bellman (1958), to compute the shortest paths (or longest paths) from a single source.

Now observe that $v_\lambda(a)$ decreases with λ. So that if there is a solution α of Relation 5 for a value λ_0, then the solution also holds for any $\lambda \geq \lambda_0$. As we consider here the existence of (α, λ) satisfying Relation 5 for each arc, such an existence can be checked for any large enough λ.

Now observe that for an arc $a = (i, j)$, if $\sigma(i)+l_{ij} > \sigma(j)$ then for $\lambda \geq \sigma(i)+l_{ij}-\sigma(j)$, $v_\lambda(a) = 1-h_{ij}$. Otherwise, $\sigma(i)+l_{ij} \leq \sigma(j) < \lambda$, so that $v_\lambda(a) = -h_{ij}$. Let us choose $\overline{\lambda} \geq \max_{(i,j)\ arc\ of\ G} \sigma(i)+l_{ij}-\sigma(j)$.

$$v_{\overline{\lambda}}(a) = \begin{cases} 1 - h_{ij} & if \quad \sigma(i) + l_{ij} > \sigma(j) \\ -h_{ij} & otherwise \end{cases}$$

So the existence of α and λ such that Relation 5 is met for each arc can be checked in polynomial time by finding whether graph G with the valuation $v\overline{\lambda}$ has no cycle of positive value. This can be checked in polynomial time using the Bellman-Ford algorithm, Cormen et al. (2009). ■

From this property, we deduce a polynomial special case of our problem:

Property 5. *If G does not contain any cycle, then the problem instance of UGF is feasible, and a feasible schedule can be computed in polynomial time.*

Proof. Let us consider any schedule σ of the super-tasks. According to the definition of the grouping constraint, this induces a schedule σ of the original tasks. Then following Property 4, as the feasibility is checked through the existence of some cycles in G, if G does not contain any cycle, it is always possible to find a retiming α and a value λ meeting Relation (5). ■

3. The special ZigBee case

The IEEE 802.15.4/ZigBee standard is the leading technology for low-cost and low-power WSNs (see IEEE (2006); ZigBee (2006); Palopoli et al. (2011)). In one of its variants, the underlying communication

topology of the devices forms an undirected tree, called the cluster-tree. Each cluster knows its neighbors in the tree. When a cluster is active, it consumes a considerable amount of energy to perform its communication tasks respecting the tree topology.

We consider that all the devices may have sensing and/or actuating capabilities, therefore, they can be sources and/or sinks of periodic message-flows, respectively. The (periodic) message-flow traverses different clusters on its routing tree determined by several source devices, and a unique sink device on the underlying cluster-tree.

In order to save the energy of the battery powered devices, the standard specifies that each cluster is active only once during the period, so that during a period, all the tasks of the message-flows passing through a cluster are grouped. In addition, the message-flows may have opposite directions, and they are constrained by the start-to-end deadlines. Consequently, not every instance has a feasible solution. In this section, we first discuss the modeling of the ZigBee problem as an instance of the UGF. Then we use the specific structure of the groups and constraints to prove that the feasibility problem can be solved in polynomial time in this case.

3.1. Modeling the ZigBee constraints

We are given a cluster-tree T, whose nodes represent the clusters and whose edges represent the logical links between them. Figure 5 shows an example of a cluster-tree consisting of nine clusters.

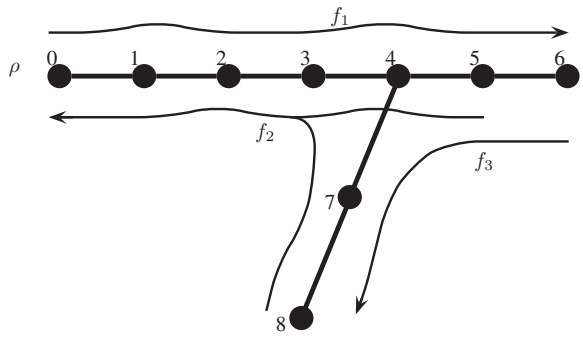


Figure 5: An example of cluster-tree T and three message-flows

We then consider a collection of message-flows. Each message-flow f is defined by a copy of a subtree of T, oriented as an in-tree, and represents the communication of data along the communication links of T from the source nodes of the message-flow to the unique sink x_f. For message-flow f, if node i belongs to the sub-tree of f, then we denote, by u_f^i, the communication task associated to node i in message-flow f. We assume the unit processing times of the communication tasks.

An example of the message-flow is given in Figure 7, where message-flow 2 has source nodes associated with clusters 5 and 8, while the sink node is associated with cluster 0.

An iteration of each message-flow will start at each period. Moreover, the energy constraints of the ZigBee standard consider that each cluster should be active once in each period.

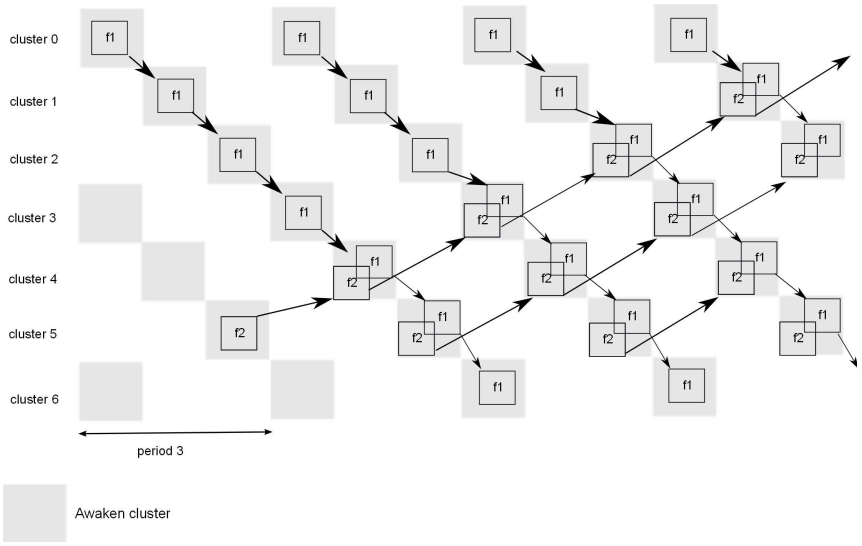


Figure 6: A periodic schedule with grouping constraints for two opposite message-flows

So tasks u_f^i for all message-flows f passing through node i belong to group i, which should be scheduled once in the period as a super-task, of the unit processing time (all together). Hence in the grouping constraint for this problem, for each message-flow f, the group of u_f^i is i, $g(u_f^i) = 0$, and for each $i \in T$, $\pi_i = 1$.

So we are looking for a grouped periodic schedule of the tasks u_f^i. We now need to model the precedence constraints associated with the data transmission and message delay.

Obviously, if j is a successor of i in the in-tree defining the message-flow f, then the k^{th} iteration of u_f^i precedes the k^{th} iteration of u_f^j. So this can be modeled by a uniform precedence constraint of length 1 and height 0.

Let us consider, for example, two opposite message-flows $f_1 = (0 \to 1 \to 2 \to 3 \to 4 \to 5 \to 6)$ and one branch of $f_2 : (5 \to 4 \to 3 \to 2 \to 1 \to 0)$, and a unit time length of each cluster activity happening once during a period. It is easy to see that if an iteration of a message-flow starts during the period $[k\lambda, (k+1)\lambda)$ then the data will be delivered to the sink node several periods later. For example, let us consider a schedule, depicted in Figure 6 with a period of $\lambda = 3$ where clusters $0, 3, 6$ are active at time 0, clusters $1, 4$ are active at time 1 and clusters $2, 5$ are active at time 2 of each period. Then, if an iteration of message-flow f_1 starts a transmission in the first period by the communication in cluster 0, then it will end the transmission in the third period by the communication in cluster 6. On the other hand, if an iteration of message-flow f_2 starts its transmission in the first period by communication in cluster 5, then it will end the transmission in the fifth period by the communication in cluster 0.

Of course, if we do not limit the time of delivery, then the periodic scheduling problem can be handled in polynomial time (see Property 5). However, in order to achieve a good response time, the start-to-end deadlines are usually settled. Here, we chose to limit the number of periods crossed by the message-flow

until its delivery.

The choice of this model with coarse-grained start-to-end delays can, of course, be discussed, but the use of cyclic scheduling tools and algorithms can usefully balance the lack of accuracy:

- For large-scale systems, we need efficient polynomial algorithms.
- Systems with a long period (to save energy), long message-flows spanning many clusters and opposite orientation of the message-flows lead to the transmission over many periods and then the relative position within the period is negligible compared to the length of the start-to-end delay.

The experiments with the scheduling tool of Ahmad & Hanzalek (2018), which enables system designers to configure all the required parameters of the IEEE 802.15.4/ZigBee cluster-tree WSNs, illustrate the efficiency of our model.

Definition 8. *For each message-flow f and each source of message-flow u_f^i we define the **period crossing constraint** associated to source u_f^i and given by the integer value c_f^i (i.e., the maximum number of crossed periods): in any periodic schedule, and any non-negative integer y, if the y^{th} iteration of u_f^i starts in the interval $[x\lambda, (x+1)\lambda)$ the y^{th} iteration of f will end during or before the interval $[(x+c_f^i)\lambda, (x+c_f^i+1)\lambda)$. We say that the periodic schedule meets the period crossing constraint for message-flow f.*

We can now associate the uniform graph G_f to each message-flow f.

3.2. The graph formulation of the ZigBee problem

1. The nodes of G_f are the tasks u_f^i for all clusters belonging to the sub-tree of f.
2. If there is a communication link $u_f^i \to u_f^j$ in the underlying sub-tree, then (u_f^i, u_f^j) is an arc of G_f with height 0.
3. If u_f^i is the sink of the message-flow f and u_f^j is a source, there is an arc from u_f^i to u_f^j with height $h(u_f^i, u_f^j) = c_f^j + 1$. Notice that this constraint can be interpreted as follows: for any integer y the time delay between the starting time of the y^{th} iteration of u_f^j and the completion of the iteration of u_f^i is at most $\lambda(c_f^i + 1)$ which is obviously implied by the period crossing constraint of message-flow f.

Uniform graph G is just the union of the message-flow sub-graphs G_f. For our example, in Figure 5, if we set $c_{f_1}^0 = 3, c_{f_2}^5 = 3, c_{f_2}^8 = 2, c_{f_3}^6 = 3$ then we get the uniform graph shown in Figure 7.

Definition 9. *A **periodic schedule is feasible** if it meets the precedence constraints associated to data transmission and the period crossing constraints.*

The following Lemma states that the period crossing constraint is met by the periodic schedules of the uniform graph defined above with an additional property.

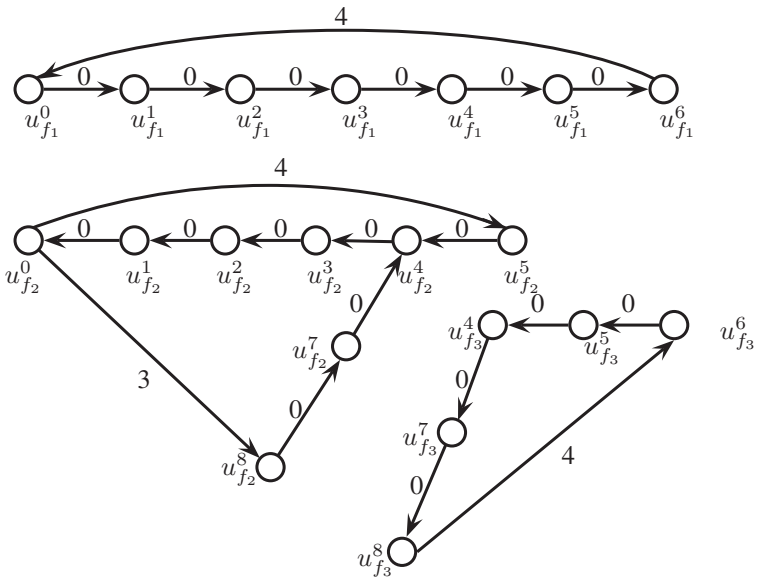


Figure 7: The uniform graph G for the message-flows in Figure 5

Lemma 1. *A feasible periodic schedule exists, if and only if, a periodic schedule of G based on retiming α exists so that for any message-flow f with source u^j_f and sink u^i_f,*

$$c^j_f + 1 + \alpha(u^j_f) - \alpha(u^i_f) > 0.$$

Proof. Consider a feasible schedule meeting the period crossing constraints for each message-flow, and α its retiming. For message-flow f with source u^j_f and sink u^i_f, consider the underlying path from node j to node i in the underlying tree T. A path in the uniform graph G, where each arc has height equal to 0, corresponds to this path.

Consider the interval $[x\lambda, (x+1)\lambda)$. We know that the iteration $y = x - \alpha(u^j_f) + 1$ of u^j_f starts in this interval. By assumption, the same iteration y of u^i_f completes before the end of the interval $[(x+c^j_f)\lambda, (x+1+c^j_f)\lambda)$. But we know that the iteration of u^i_f performed in this interval is $x+c^j_f-\alpha(u^i_f)+1$. Hence

$$x + c^j_f - \alpha(u^i_f) \geq x - \alpha(u^j_f) \quad \Rightarrow \quad \alpha(u^i_f) - \alpha(u^j_f) \leq c^j_f$$

So $c^j_f + 1 + \alpha(u^j_f) - \alpha(u^i_f) \geq 1 > 0$. q.e.d.

Conversely, let us consider a feasible periodic schedule of G such that for message-flow f, $c^j_f + 1 + \alpha(u^j_f) - \alpha(u^i_f) > 0$. Thus, $\alpha(u^i_f) - \alpha(u^j_f) \leq c^j_f$. We can then claim that $x + c^j_f - \alpha(u^i_f) \geq x - \alpha(u^j_f)$ and, thus, that the iteration $y = x-\alpha(u^j_f)+1$ of u^i_f completes before the end of interval $[(x+c^j_f)\lambda, (x+1+c^j_f)\lambda)$.

■

Notice that the specificity of the ZigBee graph is not really due to the decomposition of G into disjoint sub-trees since we can define an equivalent problem with a graph composed of disjoint cycles that are

further grouped in the grouped retiming graph $\mathcal{G}^\alpha$. Let us consider graph G'_f constructed from the message-flow sub-graph G_f as follows: Consider the message-flow f having the form of an in-tree, with sources $u_f^{i_1}, \ldots, u_f^{i_t}$. Let u_f^r be a source, and u_f^j be any node on the path from u_f^r to the sink in G_f. Then for each $r \in \{i_1, \ldots, i_t\}$ we create a copy of the path from u_f^r to the sink while using label $u_f^j(r)$ to distinguish the copy of node u_f^j in path r. Finally, we add a copy of the return arc from the copy of the sink to the copy of the source $u_f^r(r)$. So t cycles will be constructed in G'_f. Then graph G' is just the union of the message-flow sub-graphs G'_f.

For example, the in-tree message-flow 2 in Figure 5 could be represented by two paths : $5 \to 4 \to 3 \to 2 \to 1 \to 0$ with a c_f value of 3 and $8 \to 7 \to 4 \to 3 \to 2 \to 1 \to 0$ with a c_f value of 2. We can now establish that:

Lemma 2. *A grouped schedule of G satisfying the period crossing constraint exists if, and only if, a grouped schedule of G' satisfying the period crossing constraint exists.*

Proof. Notice that in G', all copies of a node of G belong to the same group and, thus, in a grouped schedule, they will merge in the same super-task. So the existence of an acyclic grouped graph $\mathcal{G}'^\alpha$ satisfying the constraints is exactly the same problem as the existence of an acyclic grouped graph $\mathcal{G}^\alpha$. ■

Thus, we can assume without loss of generality that G is a collection of disconnected cycles, each of them composed with a path from the source to the sink and a return arc.

Property 6 shows that the feasibility problem thus relies on a graph formulation of the existence of a feasible retiming.

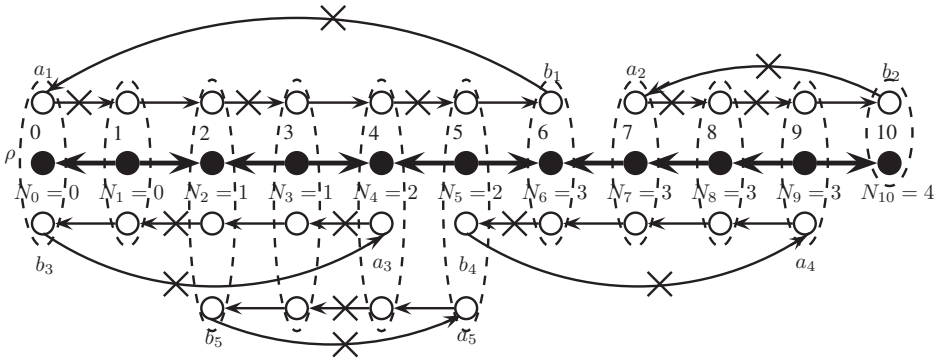
Definition 10. *α is a* **feasible retiming** *if for each message-flow f, the return arc and, at most, c_f other arcs from the cycle representing f are removed from G to build $\mathcal{R}^\alpha$ and if the grouped retiming graph $\mathcal{G}^\alpha$ is acyclic.*

Property 6. *In the ZigBee case, the feasibility problem reduces to the following graph decision problem:*

For each message-flow f, is it possible to remove the return arc and, at most, c_f other arcs from the cycle representing f (this entirely defines a retiming α) so that the grouped retiming graph $\mathcal{G}^\alpha$ obtained by merging all the nodes of the same group is acyclic?

Proof. By Lemma 1, the return arc is removed in any retiming of a periodic schedule satisfying the period crossing constraint (its height added to the difference of the retiming values is strictly positive) to get the retiming graph $\mathcal{R}^\alpha$. Property 2 yields that at most c_f other arcs should be removed from the graph to get $\mathcal{R}^\alpha$. Then the arguments of Section 2.4 achieve the proof. ■

The structure of the underlying tree T might influence the structure of the solutions. In order to have a further look into this question, we need to simplify the problem instance by assuming that the underlying tree T is a chain.



supertask	0	1	2	3	4	5	6	7	8	9	10
σ	1	0	1	0	1	0	3	2	1	0	1

Figure 8: A ZigBee-chain solution: by removing the crossed arcs we obtain the retiming graph $\mathcal{R}^\alpha$. A corresponding grouped retiming graph $\mathcal{G}^\alpha$ is represented by the filled in black nodes and bold arcs. A core schedule of the supertasks satisfying the precedence constraints of $\mathcal{G}^\alpha$ is shown.

3.3. ZigBee-chain: A simple ZigBee-like subproblem

In this case we consider that the underlying structure T is a chain of nodes numbered from 0 to $n-1$. Then, each message-flow f is completely characterized by the source index a_f, the sink index b_f and the unique period crossing constant c_f (since there is a unique source of the message-flow, we omit the upper index). The associated uniform graph is then a cycle whose nodes and arcs are:

- if $a_f < b_f$, we say f is an increasing message-flow and we define its nodes

$$u_f^{a_f}, u_f^{a_f+1}, \ldots, u_f^{b_f}$$

  with arcs (u_f^k, u_f^{k+1}) of length 1 and height 0 for all $k \in [a_f, b_f - 1]$, and an arc, called a return arc $(u_f^{b_f}, u_f^{a_f})$ with length 1 and height $c_f + 1$.

- if $a_f > b_f$, we say f is a decreasing message-flow and we define its nodes

$$u_f^{a_f}, u_f^{a_f-1}, \ldots, u_f^{b_f}$$

  with arcs (u_f^k, u_f^{k-1}) of length 1 and height 0 for all $k \in [b_f + 1, a_f]$, and a return arc $(u_f^{b_f}, u_f^{a_f})$ with length 1 and height $c_f + 1$.

Figure 8 shows a solution of the UGF based on the arc removal derived from Property 6 and the associated grouped retiming graph $\mathcal{G}^\alpha$ is depicted by the filled in black nodes and bold arrows. Figure 9 shows a grouped schedule of period $\lambda = 4$ derived from this solution.

3.3.1. Removing Bridges

Let us first assume that there is an interval $[i, j]$, where i, j are two integers with $i < j$ so that no decreasing message-flow crosses the interval, i.e., for each decreasing message-flow f, either $a_f \leq i$ or

	message-flow 1							message-flow 2			
	u_1^0	u_1^1	u_1^2	u_1^3	u_1^4	u_1^5	u_1^6	u_2^7	u_2^8	u_2^9	u_2^{10}
α	0	1	1	2	2	3	3	0	1	2	2
σ	1	0	1	0	1	0	3	2	1	0	1
s	1	4	5	8	9	12	15	2	5	8	9

	message-flow 3					message-flow 4					message-flow 5			
	u_3^4	u_3^3	u_3^2	u_3^1	u_3^0	u_4^9	u_4^8	u_4^7	u_4^6	u_4^5	u_5^5	u_5^4	u_5^3	u_5^2
α	0	1	1	2	2	0	0	0	0	1	0	0	1	1
σ	1	0	1	0	1	0	1	2	3	0	0	1	0	1
s	1	4	5	8	9	0	1	2	3	4	0	1	4	5

Figure 9: A schedule of period $\lambda = 4$ derived from the graph of Figure 8. The retiming α, core schedule σ and starting time of the first iteration of tasks s are shown. Recall that u_f^j is the task of message-flow f that passes through node j and belongs to group j.

$b_f \geq j$. We say that $[i, j]$ is a left-to-right bridge. This occurs similarly ($[i, j]$ is a right-to-left bridge) if no increasing message-flow crosses the interval, i.e., for each increasing message-flow f, either $a_f \geq j$ or $b_f \leq i$. The example shown in Figure 8 has two bridges: $[6, 7]$ is a right-to-left bridge, whereas $[9, 10]$ is a left-to-right bridge. The main idea of this subsection is to argue that the arcs crossing the bridges can never create a cycle in the grouped retiming graph $\mathcal{G}^\alpha$ of any retiming α. Hence, if we merge two consecutive nodes of a bridge we get a reduced problem, such that a feasible periodic schedule in the original problem exists, if and only if, a feasible periodic schedule in the reduced problem exists.

Let $[i, i+1]$ be a left-to-right bridge. We build a bridge-reduced problem as follows: for any increasing message-flow f such that arc (u_f^i, u_f^{i+1}) is in the message-flow, we modify f by merging node u_f^i and node u_f^{i+1} into a node u_f^i. In the ZigBee chain, node i is merged with node $i+1$, so that the numbering of the nodes is now $0, \ldots, i, i+2, \ldots, n-1$. For any message-flow f $a_f = i+1$, we change it for $a_f = i$. For any ZigBee-chain instance $\mathcal{I}$ with a left-to-right bridge, we denote it by $LRB_i(\mathcal{I})$, as the bridge-reduction described above.

Similarly if $[i, i+1]$ is a right-to-left bridge, we can, by merging the nodes associated to i and $i+1$, build a reduced instance denoted by $RLB_i(\mathcal{I})$.

Lemma 3. *For any ZigBee-chain instance $\mathcal{I}$ with a left-to-right bridge $[i, i+1]$ a periodic schedule for $\mathcal{I}$ exists, if and only if, a periodic schedule for $LRB_i(\mathcal{I})$ exists. Similarly, if $\mathcal{I}$ has a right-to-left bridge $[i, i+1]$ then a periodic schedule for $\mathcal{I}$ exists, if and only if, a periodic schedule for $RLB_i(\mathcal{I})$ exists.*

Proof. Let us consider an instance with a left-to-right bridge $[i, i+1]$. Assume that we have the retiming α such that $\mathcal{G}^\alpha$ is acyclic. Let us prove that there is no path from $i+1$ to i in $\mathcal{G}^\alpha$. Notice that in $\mathcal{G}^\alpha$, since the return arcs are removed, the only arcs are of the form $(k, k+1)$ or $(k+1, k)$. As no decreasing

message-flow crosses the bridge $[i, i+1]$, $\mathcal{G}^\alpha$ cannot contain an arc $(i+1, i)$. Hence, if we merge node i and node $i+1$ into $\mathcal{G}^\alpha$, no cycle will appear. So this will define a feasible solution for $LRB_i(\mathcal{I})$. Conversely, consider that the instance $LRB_i(\mathcal{I})$ is feasible, with a retiming α'. Then we can build α, the retiming for $\mathcal{I}$ by setting $\alpha(u_f^{i+1}) = \alpha'(u_f^i)$ and for any $k \neq i+1$ $\alpha(u_f^k) = \alpha'(u_f^k)$. Then in $\mathcal{G}^\alpha$, the arc $(i, i+1)$ will be kept, and, as argued previously, it does not induce any cycle. Moreover, the number of arcs removed from the cycles representing the message-flows to get $\mathcal{G}^\alpha$ has not increased. Hence, $\mathcal{I}$ is a feasible instance.

The same arguments can be used for the RLB_i reduction. ■

So for any problem instance with bridges, applying this reduction iteratively will lead to an equivalent instance (in terms of feasibility) without bridges. Thus, we can assume, without loss of generality, that there is no bridge.

3.3.2. A necessary and sufficient feasibility condition stated by the difference constraint system

Let us assume an instance of the feasibility problem for the chain without a bridge.

This implies that for all $k \in \{0, \ldots, n-2]\}$, an increasing message-flow AND a decreasing message-flow crossing the interval $[k, k+1]$ do exist.

So, if f is an increasing message-flow crossing this interval and f' is any decreasing message-flow crossing the interval, then **one of the arcs** (u_f^k, u_f^{k+1}) **or** $(u_{f'}^{k+1}, u_{f'}^k)$ **must be removed** from the graph in any feasible retiming, otherwise the cycle $(k, k+1, k)$ would remain in the grouped retiming graph $\mathcal{G}^\alpha$.

Assume that both are removed in a feasible retiming α. Then it is always possible to change the value of the retiming so that one of the two arcs remains in the graph. Indeed, adding an arc will not contradict the condition on the maximum number of removed arcs per cycle (see Property 6). The only problem would be if adding the arc could create a cycle in the grouped retiming graph $\mathcal{G}^\alpha$.

But, if both arcs would create a cycle, then there would be a path from k to $k+1$ (since the arc $(k+1, k)$ yields a cycle) in $\mathcal{G}^\alpha$ and similarly a path from $k+1$ to k , since the arc $(k, k+1)$ would yield a cycle. Hence, $\mathcal{G}^\alpha$ would not be acyclic. So, one of the two arcs can be added without violating the constraints.

Moreover, assume, without loss of generality, that arc (u_f^k, u_f^{k+1}) stays there and that $(u_{f'}^{k+1}, u_{f'}^k)$ is removed. Then, in any feasible retiming, if there is another decreasing message-flow f'' crossing the interval $[k, k+1]$ then arc $(u_{f''}^{k+1}, u_{f''}^k)$ has to be removed as well.

This means that the decision of keeping an arc of the form (u_f^k, u_f^{k+1}) and, thus, removing $(u_{f'}^{k+1}, u_{f'}^k)$ will apply to any message-flow crossing the same interval in the same direction. Hence, we can consider a binary choice for each interval $[k, k+1]$: to keep all arcs of the form (u_f^k, u_f^{k+1}) and to remove arcs of the form $(u_{f'}^{k+1}, u_{f'}^k)$ or vice versa: to keep $(u_{f'}^{k+1}, u_{f'}^k)$ and to remove (u_f^k, u_f^{k+1}).

We say that interval $[k, k+1]$ is "oriented from left to right" if we decide to keep the arcs of the increasing message-flows crossing the interval and remove the arcs of the decreasing message-flows. Otherwise, the interval is said to be "oriented from right to left".

For a given schedule, and each integer $j \in [1, n-1]$ let us denote, by N_j, the number of all intervals $[k, k+1]$ with $k+1 \leq j$ that are oriented from left to right. Furthermore, we state $N_0 = 0$.

We then get the following Lemma:

Lemma 4. *For any feasible retiming α, the associated variables N_j satisfy the following inequalities:*

$$\begin{cases} 0 \leq N_{k+1} - N_k \leq 1 & \forall k \in \{0, \ldots, n-2\} \\ b_f - a_f - (N_{b_f} - N_{a_f}) \leq c_f & for\ any\ increasing\ message-flow\ f \\ (N_{a_f} - N_{b_f}) \leq c_f & for\ any\ decreasing\ message-flow\ f \end{cases} \tag{6}$$

Proof. Let f be a decreasing (resp. increasing) message-flow. We know that, at most, c_f arcs are removed from the path from $u_f^{a_f}$ to $u_f^{b_f}$. If an arc (u_f^{k+1}, u_f^k) (resp (u_f^k, u_f^{k+1}) is removed the opposite arc is kept, so that the interval $[k, k+1]$ is oriented from left to right (resp from right to left). In the case of a decreasing message-flow (resp. increasing), we conclude that the number of intervals crossed by the message-flow that are oriented from left to right (resp. right to left) represents the number of removed arcs in the message-flow. But the number of intervals crossed by f that are oriented from left to right is $N_{a_f} - N_{b_f}$ for a decreasing message-flow, whereas the number of intervals crossed by an increasing message-flow f and oriented from right to left is $b_f - N_{b_f} - (a_f - N_{a_f})$. ■

We can now state the main result of this section:

Lemma 5. *If the System (6) has a solution $(N_j)_{j \in \{0,\ldots,n-1\}}$ then a feasible periodic schedule exists.*

Proof. Notice that as the system matrix of (6) is totally unimodular and all constants are integers, if the system is feasible then it has an integer solution. Hence, assuming an integer solution N_j, we notice from the first inequality that for any interval $[k, k+1)$, $N_{k+1} - N_k$ equals 0 or 1. So we define $x_k = N_{k+1} - N_k$.

Let us define the following retiming: for any increasing message-flow f and for any k such that $a_f \leq k \leq b_f$, $\alpha(u_f^k) = (k - N_k) - (a_f - N_{a_f})$ (i.e., the number of intervals between a_f and k for which an orientation from right to left is chosen). For any decreasing message-flow f and for any k such that $b_f \leq k \leq a_f$, $\alpha(u_f^k) = N_{a_f} - N_k$ (i.e., the number of intervals between a_f and k for which an orientation from left to right is chosen).

Let us remark that for any increasing message-flow f, an arc (u_f^k, u_f^{k+1}) is removed, if and only if, $\alpha(u_f^{k+1}) - \alpha(u_f^k) > 0$. But such an event occurs if $k + 1 - N_{k+1} - (k - N_k) > 0 \Leftrightarrow x_k = 0$. As $b_f - N_{b_f} - (a_f - N_{a_f}) \leq c_f$ there are at most c_f such arcs.

So the number of removed arcs on the cycle of the message-flow respects the uniform constraints, if we remove the return arc.

Similar arguments can be used for decreasing message-flows: An arc (u_f^{k+1}, u_f^k) is removed only if $\alpha(u_f^k) - \alpha(u_f^{k+1}) > 0$, i.e., if $N_{k+1} - N_k > 0$, and, thus, $x_k = 1$.

So to summarize:

- If $x_k = 0$ then the arc (u_f^k, u_f^{k+1}) is removed from any increasing message-flow f passing through k and the arc (u_f^{k+1}, u_f^k) is kept in any decreasing message-flow f passing through k.

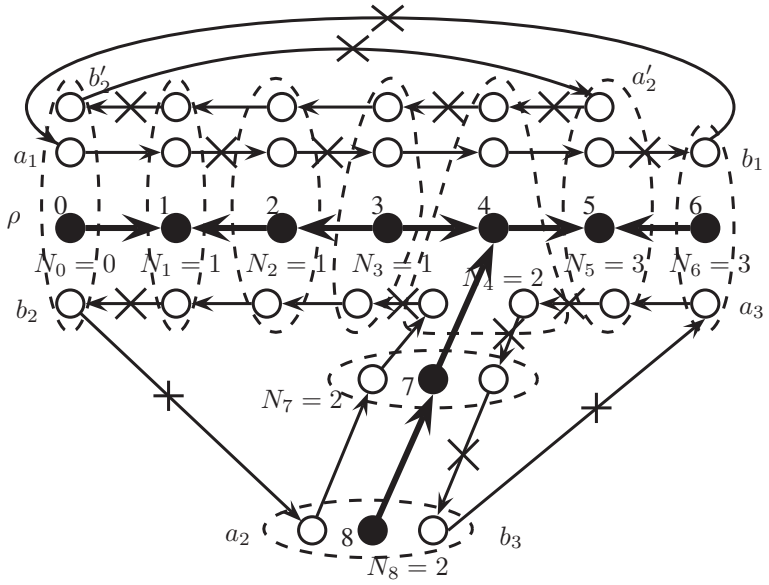


Figure 10: A solution of ZigBee: G is represented by the blank nodes and plain arcs, $\mathcal{R}^\alpha$ is obtained from G by removing the crossed arcs, $\mathcal{G}^\alpha$ is represented by the filled in nodes with bold arcs

- If $x_k = 1$ then the arc (u_f^k, u_f^{k+1}) is kept in any increasing message-flow f passing through k and the arc (u_f^{k+1}, u_f^k) is removed from any decreasing message-flow f passing through k.

Let us now build a grouped retiming graph $\mathcal{G}^\alpha$. The nodes of $\mathcal{G}^\alpha$ are $\{0, \ldots, n-1\}$. Let us show that $\mathcal{G}^\alpha$ is acyclic. First, notice that no return arc belongs to this graph so the arcs have the form $(k, k+1)$ or $(k+1, k)$. In such a graph, any elementary cycle would be composed by two such arcs.

But according to the previous properties, if $(k, k+1)$ is in the graph, then $x_k = 0$ and if $(k+1, k)$ belongs to $\mathcal{G}\alpha$, $x_k = 1$, so both arcs cannot belong to $\mathcal{G}^\alpha$. ■

Notice that System (6) is a difference constraints system that can be solved using the longest or shortest path algorithms of Cormen et al. (2009); Bellman (1958) as mentioned above. The complexity is polynomial in the number of nodes n and the number of message-flows z ($O(nm)$ where m is the number of arcs, which depends on z and n). At this stage, we could consider that the ZigBee chain can be solved in polynomial time. Unfortunately, the input of the ZigBee-chain problem given by $(a_f, b_f, c_f)_{message-flows\ f}$ can be encoded in $O(log(n) * z)$ (where z is the number of message-flows), since the graph of the chain is not part of the input. Thus, the complexity is not a polynomial of the input size (it is a polynomial of the number of nodes). However, the system can be extended in order to handle the general ZigBee setting (where the network topology is part of the input) that will be solved in polynomial time.

3.4. The ZigBee problem feasibility can be solved in polynomial time

In the original ZigBee problem, the input comprises the tree T, which has size $O(n)$ since the topology needs to be encoded in the input.

Among the properties studied for the ZigBee-Chain, the following properties still hold:

1. The bridges can still be merged so that we can assume no bridge (i.e., no edge of the underlying subtree is crossed by message-flows of the same direction only).
2. If two message-flows have opposite directions on one edge of the tree, then, of course, in the retiming, one of the arcs will be removed and the other kept; otherwise a cycle would appear in the grouped retiming graph. Moreover, the decision made on one arc applies to all the message-flows crossing this arc in the same direction.

Notice that as the underlying structure is a tree and not a chain, there is no real "left to right" or "right to left" natural orientation. So, in the rest of the paper, we consider an arbitrary underlying orientation of the edges of the tree T so that it forms an out-tree $\hat{T}$ with some root ρ. For any arc (a, b) of this out-tree $\hat{T}$, and any feasible grouped retiming graph $\mathcal{G}^\alpha$, if (a, b) is in $\mathcal{G}^\alpha$, we say that **the arc** (a, b) **is topological**, whereas if (b, a) is in $\mathcal{G}^\alpha$, we say the arc (a, b) is **inverted**.

From these properties, it is possible to formulate inequalities equivalent to System (6) that gives a necessary and sufficient condition of feasibility as follows:

A message-flow f, in the form of a chain from the source node a_f to the sink node b_f can have the following forms:

- f is an increasing message-flow if there is a oriented path from a_f to b_f in the given out-tree $\hat{T}$.
- f is a decreasing message-flow if there is a oriented path from b_f to a_f in the given out-tree $\hat{T}$.
- Otherwise, f is said to be a bipolar message-flow. According to the tree structure, and as f is a subchain of the non oriented tree T, a node z_f in T crossed by f exists so that in the given out-tree $\hat{T}$, there is a path from z_f to a_f and a path from z_f to b_f. So f is "decreasing" from a_f to z_f and "increasing" from z_f to b_f.

We denote for node j of the tree by $\nu(j)$, the number of arcs on the unique path from root ρ to node j in the orientation given by $\hat{T}$ (i.e., the distance from the root). This value will replace the numbering of nodes used in the Zigbee-chain.

We then have the following result:

Theorem 2. *A feasible periodic schedule exists, if and only if, the following system has an integer solution N:*

$$\begin{cases} 0 \leq N_j - N_i \leq 1 & \forall (i,j) \text{ arc of } \hat{T} \\ \nu(b_f) - \nu(a_f) - (N_{b_f} - N_{a_f}) \leq c_f & \text{for any increasing message} - \text{flow } f \\ N_{a_f} - N_{b_f} \leq c_f & \text{for any decreasing message} - \text{flow } f \\ N_{a_f} + \nu(b_f) - \nu(z_f) - N_{b_f} \leq c_f & \text{for any bipolar message} - \text{flow } f \end{cases} \tag{7}$$

The system can be checked in polynomial time.

Proof. Now, similarly to the ZigBee chain case, we first prove that if a schedule exists, then it defines the orientation of the edges of T and, thus, we can compute the associated value N_j that satisfies System (7).

Let us consider a feasible schedule, its retiming α, retiming graph $\mathcal{R}^\alpha$ and grouped graph $\mathcal{G}^\alpha$. We define for any node j of the tree, N_j to be the number of topological arcs of $\mathcal{G}^\alpha$ along the chain from ρ to j (i.e., the arcs for which the orientation in $\mathcal{G}^\alpha$ coincides with the orientation of $\hat{T}$).

The proof for the increasing or decreasing message-flows is similar to the one of the ZigBee-chain, so we shall skip it here, and focus on the case of the bipolar message-flow f with its intermediate node common ancestor z_f of a_f and b_f in $\hat{T}$.

We need to express the constraint saying that the number of arcs of the message-flow f that are removed is not greater than c_f. Obviously, an arc of the message-flow is removed by the retiming if its chosen orientation (topological or inverted) is not the one given by the message-flow (otherwise, the arc of the message-flow would induce a cycle). So, for f, this is the number of topological arcs along the chain from a_f to z_f (since, in the out-tree, we have a path from z_f to a_f, opposite to the direction of this part of f) plus the number of inverted arcs along the path from z_f to a_f.

The first member is $N_{a_f} - N_{z_f}$, and the second is $\nu(b_f) - N_{b_f} - (\nu(z_f) - N_{z_f})$. So we get the following number of removed arcs on the $a_f \rightarrow z_f \rightarrow b_f$ path $(N_{a_f} - N_{z_f}) + \nu(b_f) - \nu(z_f) - (N_{b_f} - N_{z_f}) = N_{a_f} + \nu(b_f) - \nu(z_f) - N_{b_f}$.

Hence, this number is less than c_f, so that System (7) holds.

Conversely, we show that the solution of System (7) provides a feasible retiming.

We define:

- If f is an increasing message-flow and k is a node through which f is passing: $\alpha(u_f^k) = \nu(k) - N_k - (\nu(a_f) - N_{a_f})$
- If f is a decreasing message-flow and k is a node through which f is passing: $\alpha(u_f^k) = N_{a_f} - N_k$
- If f is a bipolar message-flow with an intermediate node z_f and k is a node through which f is passing in its decreasing part: $\alpha(u_f^k) = N_{a_f} - N_k$. If k is in the increasing part, then $\alpha(u_f^k) = N_{a_f} - N_{z_f} + \nu(k) - N_k - (\nu(z_f) - N_{z_f})$

We now have to check that the number of arcs removed from the cycle associated to the bipolar message-flow f in $\mathcal{R}^\alpha$ is not greater than c_f (the proof for the increasing or decreasing message-flow is similar to the ZigBee-chain case). Let (a, b) be an arc of a bipolar message-flow f in G with height 0. If the arc (a, b) belongs to $\hat{T}$ then (a, b) belongs to its increasing part, then $\alpha(b) - \alpha(a) = N_{a_f} - N_{z_f} + \nu(b) - N_b - (\nu(z_f) - N_{z_f}) - (N_{a_f} - N_{z_f} + \nu(a) - N_a - (\nu(z_f) - N_{z_f}))$. But $\nu(b) - \nu(a) = 1$. So $\alpha(b) - \alpha(a) = 1 - (N_b - N_a) \in \{0, 1\}$. As we know that $0 \leq N_b - N_a \leq 1$, the arc (a, b) is removed to get $\mathcal{R}^\alpha$ only if $\alpha(b) - \alpha(a) = 1$. Hence, the number of removed arcs in the increasing part is $\alpha(u_f^{b_f}) - \alpha(u_f^{z_f}) = \nu(b_f) - \nu(z_f) - (N_{b_f} - N_{z_f})$. Similarly, let us assume that (b, a) belongs to $\hat{T}$, then $\alpha(b) - \alpha(a) = N_{a_f} - N_b - (N_{a_f} - N_a) = N_a - N_b$. As we know that $0 \leq N_a - N_b \leq 1$, the arc (a, b) is removed to get $\mathcal{R}^\alpha$ only if $\alpha(b) - \alpha(a) = 1$. Hence, the number of removed arcs from the decreasing part is $\alpha(u_f^{z_f}) - \alpha(u_f^{a_f}) = N_{a_f} - N_{z_f}$. So the whole number

of removed arcs of height 0 is: $\nu(b_f) - \nu(z_f) - (N_{b_f} - N_{z_f}) + N_{a_f} - N_{z_f} = N_{a_f} + \nu(b_f) - \nu(z_f) - N_{b_f} \leq c_f$ from System (7).

Let us now consider the return arc $(u_f^{b_f}, u_f^{a_f})$ of the message-flow. Let us compute $\alpha(u_f^{a_f}) - \alpha(u_f^{b_f}) = \alpha(u_f^{a_f}) - \alpha(u_f^{z_f}) + \alpha(u_f^{z_f}) - \alpha(u_f^{b_f}) \geq -c_f$. As the height of this arc is $c_f + 1$ in G, the chosen retiming yields $c_f + 1 + \alpha(u_f^{a_f}) - \alpha(u_f^{b_f}) > 0$ so that the return arc is removed to get $\mathcal{R}^\alpha$.

Finally, we prove that the grouped retiming graph $\mathcal{G}^\alpha$ is acyclic. As all the return arcs are removed to get $\mathcal{G}^\alpha$, the structure of $\mathcal{G}^\alpha$ is the orientation of the underlying tree given by the retiming. So there cannot be any other cycles than those linking the two adjacent nodes in T, which cannot occur. Indeed, let (a, b) be an arc of $\hat{T}$ and the two message-flows f and f' crossing the arc, so that (u_f^a, u_f^b) is an arc of f and $(u_{f'}^b, u_{f'}^a)$ is an arc of f'. According to the definition of α, $\alpha(u_f^b) - \alpha(u_f^a) = 1 + N_b - N_a$ whereas $\alpha(u_f^b) - \alpha(u_f^a) = N_a - N_b$ clearly, both values cannot equal 1, so one of the two arcs vanishes in $\mathcal{G}^\alpha$.

■

Corollary 1. *The existence of a schedule for the ZigBee problem can be determined in polynomial time*

Proof. The matrix of System (7) is totally unimodular, so that the integer solution dominates. It is a difference constraints system that can be solved by the Bellman-Ford algorithm Cormen et al. (2009). Hence, the feasibility of the system can be checked in polynomial time. ■

NOTE: If the problem is not feasible then probably some constraints are too hard. We suggest that System (7) could be extended to handle an optimization problem (8): starting from the minimal requirements for the c_f values, we could add a variable part g for all message-flows and, thus, formulate the following integer linear program, so that it can be solved in polynomial time by the interval bisection method on g combined with the LP for the feasibility of a fixed g :

$$\begin{cases} Min \quad g & \\ 0 \leq N_j - N_i \leq 1 & \forall (i,j)\ arc\ of\ \hat{T} \\ \nu(b_f) - \nu(a_f) - (N_{b_f} - N_{a_f}) \leq c_f + g & for\ any\ increasing\ message - flow\ f \\ N_{a_f} - N_{b_f} \leq c_f + g & for\ any\ decreasing\ message - flow\ f \\ N_{a_f} + \nu(b_f) - \nu(z_f) - N_{b_f} \leq c_f + g & for\ any\ bipolar\ message - flow\ f \end{cases} \tag{8}$$

4. UGF is NP-complete

The grouping constraints might be considered as a special case of the core precedences introduced in Hanzalek & Hanen (2015). They can be easily embedded in an ILP formulation, even with additional resource constraints and timing considerations. While the existence of a feasible schedule with core precedences has been proven NP-hard by Hanzalek & Hanen (2015), the complexity of UGF remained open. In this section, we use the graph formulation of UGF to prove the NP-completeness of the problem with a reduction from $3 - SAT$.

Theorem 3. *UGF is NP-Complete*

The proof of this theorem relies on several definitions and lemmas. After proving its membership to the NP class, we define a reduction from 3-SAT. Then we will establish that when we are given a satisfiable instance of 3-SAT then the associated instance of UGF does have a periodic schedule. Conversely, we start from a periodic schedule of an instance of UGF to get a solution of 3-SAT.

Lemma 6. *UGF belongs to NP.*

Proof. If there is a feasible schedule, then following Lemma 3, the definition of the retiming (whose size is polynomial) will allow one to check the feasibility in polynomial time and define a feasible grouped schedule. ■

4.1. A polynomial reduction

Definition of $3-SAT$

Let us consider an instance of the $3-SAT$ problem. We are given n binary variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$. Each clause is defined by 3 literals. A literal is either an x_i or its complement $\overline{x_i}$.

The decision problem $3-SAT$ can be formulated as follows:

- Do binary values of the variables exist so that, in each clause, at least one literal is true?

This problem can be illustrated by the following example with 4 variables and 3 clauses: $C_1 = \{x_1, x_2, \overline{x_3}\}$, $C_2 = \{x_2, \overline{x_3}, x_4\}$, $C_3 = \{\overline{x_1}, x_3, \overline{x_4}\}$

For this example, $x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 0$ provides the satisfiability of this system.

Let us associate a $3-SAT$ instance with an instance of UGF as illustrated in Figure 11.

The uniform graph

We build a uniform graph associated to an instance of 3-SAT as follows: all the tasks defined below have unit processing times, and all the arcs have unit length.

Let us define, for each variable x_i of the 3-SAT instance, two tasks denoted by x_i and $\overline{x_i}$ with an arc from x_i to $\overline{x_i}$ with height 0, and an arc from $\overline{x_i}$ to x_i with height 1. These arcs and nodes are called *decision cycles*

Moreover, we consider $3n$ additional nodes $z_1, \ldots, z_n, \overline{z_1}, \ldots, \overline{z_n}, y_1, \ldots, y_n$ with the following arcs: $\forall k \in 1, \ldots n-1$, there are four arcs (z_{k+1}, y_k), $(\overline{z_{k+1}}, y_k)$, (y_k, z_k), $(y_k, \overline{z_k})$ with height 0. Moreover, we add two arcs $(z_1, y_n), (\overline{z_1}, y_n)$ with height 1. The subgraph composed with the nodes $z_i, \overline{z_i}, y_i$ and the arcs described above is called the *ordering structure*

Now, let us define, for each clause C_j six tasks c_j^0, c_j^1, c_j^2, c_j^3, c_j^4, c_j^5 linked as a cycle : for each $k \in 0, \ldots, 4$ there is an arc (c_j^k, c_j^{k+1}) with height 0. Moreover, there is an arc (c_j^5, c_j^0) with height 4.

The cycle is called a *Clause cycle*. Moreover, among the cycle itself, the arcs starting from c_j^0, c_j^2, c_j^4 are called *literal arcs* and the other ones are called *sequence arcs*.

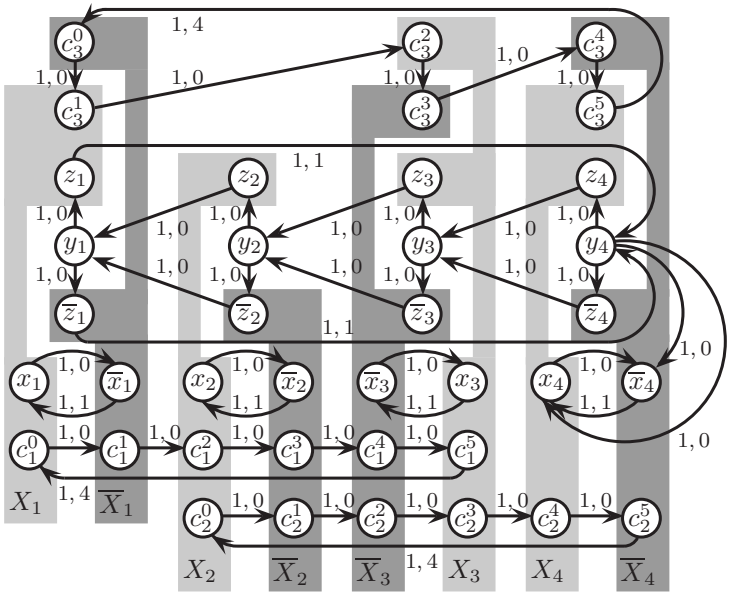


Figure 11: A uniform graph with groups for the 3-SAT example with 4 variables and 3 clauses.

Groups

There are $3n$ groups. For each variable x_i of the SAT instance, we define three groups $X_i, \overline{X_i}, Y_i$. Group $Y_i = \{y_i\}$.

Nodes x_i, z_i belong to group X_i, nodes $\overline{x_i}, \overline{z_i}$ belong to group $\overline{X_i}$.

Consider a clause C_j whose literals are sorted in increasing order of the index of variables. We consider that the literals are indexed by $0, 1, 2$. If the k^{th} literal is x_i then $c_j^{2k} \in X_i$, and $c_j^{2k+1} \in \overline{X_i}$. If the k^{th} literal is $\overline{x_i}$ then $c_j^{2k} \in \overline{X_i}$, and $c_j^{2k+1} \in X_i$.

4.2. Defining a feasible grouped periodic schedule from an instantiation of variables

Lemma 7. *If the instance of the $3-SAT$ problem is satisfiable, then there is a grouped periodic schedule of G.*

Proof. To prove this Lemma, we first define the retiming α associated to the instantiation of the variables. We then show that graph $\mathcal{R}^\alpha$ is acyclic, and that the number of removed arcs satisfy the constraints induced by the uniform graph. In the third part, we then build the grouped retiming graph $\mathcal{G}^\alpha$ and prove it is still acyclic. According to the graph formulation of UGF, this is sufficient to build a periodic schedule based on this retiming.

Let us consider an instantiation $\hat{x_1}, \ldots, \hat{x_n}$ of the variables such that each clause is satisfiable. To this instantiation, we associate the following retiming:

$$\forall i \in \{1, \ldots, n\}, \quad \begin{cases} \alpha(z_i) = \alpha(\overline{z_i}) = \alpha(y_i) = 0 & \\ \alpha(x_i) = \alpha(\overline{x_i}) = 0 & if \quad \hat{x_i} = 1 \\ \alpha(x_i) = 0, \alpha(\overline{x_i}) = 1 & if \quad \hat{x_i} = 0 \end{cases}$$

Notice that in $\mathcal{R}^\alpha$, the only arcs vanishing from the ordering structure (based on nodes $z_i, \overline{z_i}, y_i$) are those of height 1, namely $(z_1, y_n), (\overline{z_1}, y_n)$. And the subgraph of $\mathcal{R}^\alpha$ supported by the ordering structure is acyclic. Moreover, any remaining arc (u, v) so that $u \in \{z_i, \overline{z_i}, y_i\}, v \in \{z_l, \overline{z_l}, y_l\}$ satisfies $l \leq i$.

Now, if the variable $\hat{x_i}$ is true then the arc $(x_i, \overline{x_i})$ remains in $\mathcal{R}^\alpha$ while $(\overline{x_i}, x_i)$ vanishes, and if $\hat{x_i}$ is false then $(\overline{x_i}, x_i)$ remains in $\mathcal{R}^\alpha$ while $(x_i, \overline{x_i})$ vanishes.

For each clause C_j, the retiming of the clause nodes is defined as follows:

- $\alpha(c_j^0) = 0$.
- $\alpha(c_j^2) = \alpha(c_j^1) + 1, \quad \alpha(c_j^4) = \alpha(c_j^3) + 1$.
- If the literal $k \in \{0, 1, 2\}$ is true:
  - Then $\alpha(c_j^{2k}) = \alpha(c_j^{2k+1})$
  - Otherwise $\alpha(c_j^{2k+1}) = \alpha(c_j^{2k}) + 1$.

These equations define all the retiming values for the clause nodes. They are defined so that there is only one remaining sequence arc: (c_j^5, c_j^0) in the graph $\mathcal{R}^\alpha$, and at least one remaining literal arc corresponding to the true literal of the clause. As the number of arcs of the clause cycle is 6, if at least two arcs remain in $\mathcal{R}^\alpha$, then at most 4 arcs vanish, which is the constraint induced by this cycle (see Property 2). Notice that since the literals of a clause are sorted in an increasing order of the index, if $c_j^5 \in X_i \cup \overline{X_i}$ then $c_j^0 \in X_l \cup \overline{X_l}$ with $l < i$.

According to this retiming, graph $\mathcal{R}^\alpha$ is acyclic, since at least one arc vanishes from every cycle of G.

Let us now consider the grouped retiming graph $\mathcal{G}^\alpha$, in which all the nodes of a group are merged into a node. Let us prove that $\mathcal{G}^\alpha$ does not contain any cycle.

The nodes of $\mathcal{G}^\alpha$ are the groups $X_i, \overline{X_i}, Y_i$.

Let us consider the group X_i of some literal x_i.

If $\hat{x_i} = 1$ then group $\overline{X_i}$ is a successor of X_i in $\mathcal{G}^\alpha$. There is no arc from $\overline{X_i}$ to X_i. Indeed, the only such possible arc would be a literal arc (c_j^k, c_j^{k+1}) of a clause C_j. According to the definition of the literal arcs, if $c_j^k \in \overline{X_i}$ then the k^{th} literal of clause C_j is $\overline{x_i}$. But the literal arc remains in $\mathcal{R}^\alpha$ only if $\hat{x_i} = 0$.

Notice that if $\hat{x_i} = 0$ the same arguments about the clause arcs can be stated so that in this case X_i is a successor of $\overline{X_i}$ in $\mathcal{G}^\alpha$ and there is no arc from X_i to $\overline{X_i}$.

Now assume that there is a cycle in $\mathcal{G}^\alpha$. There would, thus, be an arc (u, v) so that $u \in \{X_i, \overline{X_i}, Y_i\}$ and $v \in \{X_l, \overline{X_l}, Y_l\}$ with $i < l$. This arc would be issued from an arc of $\mathcal{R}^\alpha$. But in $\mathcal{R}^\alpha$, this cannot be a clause arc (neither a literal nor a sequence arc) nor an arc from the ordering structure. So, it does not exist. ■

4.3. Building a feasible instantiation from a feasible grouped periodic schedule

Let us consider a grouped periodic schedule of G, with its core σ, retiming α and corresponding graphs $\mathcal{R}^\alpha$ and $\mathcal{G}^\alpha$.

Lemma 8. *For any variable i, either $(x_i, \overline{x_i})$ or $(\overline{x_i}, x_i)$ is an arc of $\mathcal{R}^\alpha$.*

Proof. According to Property 2, as the cycle $(x_i, \overline{x_i})$ is of height 1, exactly one arc will be removed from G in $\mathcal{R}^\alpha$. ■

Lemma 9. *In $\mathcal{R}^\alpha$, all the arcs from the ordering structure are present, except a pair of arcs either from node y_{k_0+1} or to node y_{k_0}. Moreover, the tasks $y_1, \ldots, y_n$ are scheduled in the core according to a circular permutation of the sequence $(y_n, y_{n-1}, \ldots, y_1)$. Consequently, in $\mathcal{G}^\alpha$, if $u \in \{X_i, \overline{X_i}\}$ and $v \in \{X_l, \overline{X_l}\}$ with $i < l$ then :*

$$\begin{cases} if \quad k_0 \geq l \text{ or } k_0 + 1 < i & v \to u & \text{in } \mathcal{G}^\alpha \\ if \quad l - 1 > k_0 \geq i & u \to v & \text{in } \mathcal{G}^\alpha \\ if \quad i = k_0 + 1 \text{ or } l = k_0 + 1 & u \to v \text{ or } v \to u & \text{according to the removed arcs} \end{cases} \tag{9}$$

Proof. According to Property 2, as the height of the cycle $(y_n, z_n, y_{n-1}, z_{n-1}, \ldots, y_1, z_1)$ is 1, exactly one arc will be removed from this cycle in $\mathcal{R}^\alpha$. Assume that, in the cycle $\{y_n, z_n, y_{n-1}, z_{n-1}, ...y_1, z_1\}$, the arc (y_{k_0+1}, z_{k_0+1}) is removed. Then, either the arc $(y_{k_0+1}, \overline{z_{k_0+1}})$ or the arc $(\overline{z_{k_0+1}}, y_{k_0})$ is also removed. Otherwise, there would be a cycle in $\mathcal{R}^\alpha$: $\overline{z_{k_0+1}}, y_{k_0}, \ldots, z_2, y_1, z_1, y_n, \ldots, y_{k_0+1}, \overline{z_{k_0+1}}$.

Similarly, if the arc (z_{k_0+1}, y_{k_0}) is removed, then either $(y_{k_0+1}, \overline{z_{k_0+1}})$ or $(\overline{z_{k_0+1}}, y_{k_0})$ is removed, otherwise, there would be the following cycle in $\mathcal{R}^\alpha$:

$$\overline{z_{k_0+1}}, y_{k_0}, \ldots, z_2, y_1, z_1, y_n, \ldots, y_{k_0+1}, \overline{z_{k_0+1}}.$$

The remaining arcs induce a path, starting in both cases by y_{k_0} and following the circular permutation of the indexes: $y_{k_0}, \ldots, y_1, y_n, \ldots, y_{k_0+1}$.

Now, let us consider the grouped graph $\mathcal{G}^\alpha$, and u and v nodes so that $u \in \{X_i, \overline{X_i}\}$ and $v \in \{X_l, \overline{X_l}\}$ with $i < l$. Notice that if $v = X_l$ (resp. $\overline{X_l}$) then as z_l (resp. $\overline{z_l}$) also belongs to group X_l (resp. $\overline{X_l}$), then if $k_0 \geq l$ the arc (z_l, y_{l-1}) (resp. $(\overline{z_l}, y_{l-1})$) remains in $\mathcal{R}^\alpha$, and induces an arc (v, Y_{l-1}) in $\mathcal{G}^\alpha$. The path $v, Y_{l-1}, \ldots, Y_i$ (resp $\overline{X_l} \ldots$) remains in the graph, and combined with arc (Y_i, u) will form a path from v to u.

If, now, $k_0 + 1 < i$, then the ordering structure will induce a path in $\mathcal{G}^\alpha$ starting from v, then Y_l, then a path to Y_i and the arc (Y_i, u). Hence $v \to u$.

Consider the case $i = k_0+1$. Then, as shown previously, in $\mathcal{R}^\alpha$: one of the two arcs $(y_{k_0+1}, z_{k_0+1}), (z_{k_0+1}, y_{k_0})$ is removed, while one of the two arcs $(y_{k_0+1}, \overline{z_{k_0+1}}), (\overline{z_{k_0+1}}, y_{k_0})$ is removed. This implies that in $\mathcal{G}^\alpha$ either (Y_{k_0+1}, X_{k_0+1}) or (X_{k_0+1}, Y_{k_0}) is removed, and similarly either $(Y_{k_0+1}, \overline{X_{k_0+1}})$ or $(\overline{X_{k_0+1}}, Y_{k_0})$ is removed.

Now, if $u = X_{k_0+1}$, and if (Y_{k_0+1}, X_{k_0+1}) is removed, we have, in $\mathcal{G}^\alpha$, an arc (u, Y_{k_0}) then a path from Y_{k_0} to Y_l and an arc (Y_l, v) so that $u \to v$. If $u = X_{k_0+1}$, and if (X_{k_0+1}, Y_{k_0}) is removed, then we have in $\mathcal{G}^\alpha$ an arc from v to Y_l, a path from Y_l to Y_{k0+1} and an arc (Y_{k_0+1}, X_{k_0+1}) which together form a path from v to u. Similar arguments can be used considering $u = \overline{X_{k_0+1}}$.

Consider the case where $l = k_0 + 1$ and $v = X_l$. Then, if (X_l, Y_{k_0}) is in $\mathcal{G}^\alpha$, we can build a path $v, Y_{k_0}, \ldots, Y_{i+1}, u$. Otherwise, if (Y_l, X_l) is in $\mathcal{G}^\alpha$, we can build a path $u, Y_i, \ldots, Y_1, \ldots, Y_l, v$. Similar arguments can be used if $v = \overline{X_l}$

Finally, if $i \leq k_0 < l - 1$ we can constitute a path from u to v by combining (u, Y_i) with a path from Y_i to Y_l following the ordering structure, and an arc (Y_l, v). ■

In the graph $\mathcal{G}^\alpha$, let $U_k = X_k \cup \overline{X_k}$. Let us denote, for two nodes, u, u' the relationship $u \to u'$ if there is a path from u to u' in $\mathcal{G}^\alpha$.

Lemma 10. *In $\mathcal{R}^\alpha$, for each clause C_j, there is, at most, one remaining sequence arc, and, at least, one remaining literal arc.*

Proof. Notice that as the height of the clause cycle is 4, according to Property 2, at most 4 arcs are removed from G to get $\mathcal{R}^\alpha$. Now, consider the index of the variables of the clause $k_1 < k_2 < k_3$. The sequence arcs induce the arcs $(u', v), (v', w), (w', u)$ in $\mathcal{G}^\alpha$, with $u, u' \in \{X_{k_1}, \overline{X_{k_1}}\}$, $v, v' \in \{X_{k_2}, \overline{X_{k_2}}\}$,$w, w' \in \{X_{k_3}, \overline{X_{k_3}}\}$.

But, according to Lemma 9 and depending on the situation of the break in the ordering structure k_0, we necessarily have two of the three arcs that induce a cycle with a path from the ordering structure:

- if $k_0 + 1 < k_1$ then System (9) says that $w \to v'$ and $v \to u'$.
- if $k_0 + 1 = k_1$ then $w \to v'$. Notice that if $u = X_{k_1}$ then $u' = \overline{X_{k_1}}$ (or conversely). According to System (9), $w \to v'$ and if $u' \to v$ then $u \to w'$ so that arc (w', u) induces a cycle, whereas if $v \to u'$ the arc (u', v) induces a cycle.
- if $k_1 \leq k_0$ and $k_0 + 1 < k_2$ then System (9) says that $w \to v'$ and $u \to w'$.
- similar arguments hold for other cases.

Thus, there cannot be two sequence arcs remaining in $\mathcal{G}^\alpha$. As there are 6 arcs in the clause cycle (3 literal arcs, and 3 sequence arcs), and as there is, at most, one remaining sequence arc there is, at least, one remaining literal arc (otherwise 5 arcs would have been removed) ■

Lemma 11. *A solution of the associated $3-SAT$ instance can be constructed from any grouped periodic schedule of G.*

Proof. Assume that a grouped periodic schedule of G has been found, and consider its retiming α. According to Lemma 8, there will be, for each variable index i, either the arc $(X_i, \overline{X_i})$ or the arc $(\overline{X_i}, X_i)$ remains in $\mathcal{G}^\alpha$. Let us set:

$$\begin{cases} x_i = 1 \quad if \quad (X_i, \overline{X_i}) \in \mathcal{G}^\alpha \\ x_i = 0 \quad if \quad (\overline{X_i}, X_i) \in \mathcal{G}^\alpha \end{cases}$$

We claim that this is a true assignment for each clause. Indeed, let us consider a clause. We know, from Lemma 10 that there is at least one remaining literal arc in the associated clause cycle. Assume that the corresponding literal is x_j (resp. $\overline{x_j}$). The literal arc links in $\mathcal{G}^\alpha$ the nodes: $(X_j, \overline{X_j})$ (resp. $(\overline{X_j}, X_j)$).

So, since $\mathcal{G}^\alpha$ does not contain any cycle, the arc remaining from the variable arcs cannot be $(\overline{X_j}, X_j)$ (resp. $(X_j, \overline{X_j})$). So $x_j = 1$ in the instantiation (resp. $\overline{x_j} = 1$). So the clause is true. ■

5. Conclusion

This paper was motivated by the study of cyclic scheduling problems induced by the ZigBee standard for Wireless Sensor Networks.

We introduced a new grouping constraint in a cyclic scheduling framework, and our contribution is both to the cyclic scheduling theory, where such constraints had never been introduced, and to the particular problem induced by the ZigBee standard.

In the case of the general problem UGF of checking the existence of a grouped periodic schedule, we proposed a graph formulation using a retiming graph and a grouped retiming graph. UGF has been shown to be NP-complete even if unit processing times are assumed and no additional resource constraints are considered.

However, the graph formulation of the problem allowed us to propose a model for the ZigBee case by formulating the start-to-end deadlines of the message-flows in terms of the number of periods crossed by a message-flow. We use the particular tree structure of the groups to provide a polynomial solution to the problem.

The mathematical properties established in this paper are important, among other things, for Wireless Sensor Networks with realistic flows, resource constraints (i.e., multiple collision domains) and timing constraints. In Ahmad & Hanzalek (2018) the application context of the results was investigated and an efficient configuration tool calculating the parameters of the ZigBee network was designed. Heuristic algorithms based on this work are able to find near-optimal configuration parameters for large Wireless Sensor Networks with thousands of clusters and multiple collision domains.

Beyond feasibility, optimization problems might be addressed in the future, using the tools we developed in the paper. Minimizing the period length of a periodic grouped schedule (when it exists) should be our next cyclic scheduling challenge.

Acknowledgements: This work was funded by the EU Structural funds and the Ministry of Education of the Czech Republic within the project Cluster 4.0 (number CZ.02.1.01/0.0/0.0/16_026/0008432) and by by the Technology Agency of the Czech Republic under the Centre for Applied Cybernetics TE01020197.

References:

Ahmad, A., & Hanzalek, Z. (2018). An energy efficient schedule for IEEE 802.15.4/ZigBee cluster tree WSN with multiple collision domains and period crossing constraint. *IEEE Transactions on Industrial Informatics*, *14*, 12–23.

Alcaide, D., Chu, C., Kats, V., Levner, E., & Sierksma, G. (2007). Cyclic multiple-robot scheduling with time-window constraints using a critical path approach. *European Journal of Operational Research*, *177*, 147–162.

Bellman, R. (1958). On a routing problem. *Quarterly of Applied Mathematics*, *16*, 87–90.

Benabid, A., & Hanen, C. (2011). Worst case analysis of decomposed software pipelining for cyclic unitary RCPSP with precedence delays. *Journal of Scheduling*, *14*, 511–522.

Bocewicz, G., Nielsen, P., Banaszak, Z. A., & Wójcik, R. (2017). An analytical modeling approach to cyclic scheduling of multiproduct batch production flows subject to demand and capacity constraints. In *ISAT (2)* (pp. 277–289). Springer volume 656 of *Advances in Intelligent Systems and Computing*.

Bodin, B., Kordon, A. M., & de Dinechin, B. D. (2016). Optimal and fast throughput evaluation of CSDF. In *Proceedings of the 53rd Annual Design Automation Conference, DAC 2016, Austin, TX, USA, June 5-9, 2016* (pp. 160:1–160:6).

Bonfietti, A., Lombardi, M., Benini, L., & Milano, M. (2011). A constraint based approach to cyclic RCPSP. In *Principles and Practice of Constraint Programming CP 2011. Lecture Notes in Computer Science, vol 6876. Springer, Berlin, Heidelberg* (pp. 130–144).

Brucker, P., & Kampmeyer, T. (2005). Tabu search algorithms for cyclic machine scheduling problems. *Journal of Scheduling*, *8*, 303–322. 10.1007/s10951-005-1639-4.

Calland, P.-Y., Darte, A., & Robert, Y. (1998). Circuit retiming applied to decomposed software pipelining. *IEEE Trans. Parallel Distrib. Syst.*, *9*, 24–35. doi:`http://dx.doi.org/10.1109/71.655240`.

Che, A., Feng, J., Chen, H., & Chu, C. (2015). Robust optimization for the cyclic hoist scheduling problem. *European Journal of Operational Research*, *240*, 627–636.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms, Third Edition*. (3rd ed.). The MIT Press.

Darte, A., & Huard, G. (2000). Loop shifting for loop compaction. *International Journal of Parallel Programming*, *28*, 499–534.

Dupont de Dinechin, B., & Munier-Kordon, A. (2014). Converging to periodic schedules for cyclic scheduling problems with resources and deadlines. *Computers and Operations Research*, *51*, 227 – 236.

Feng, J., Chu, C., & Che, A. (2018). Cyclic jobshop hoist scheduling with multi-capacity reentrant tanks and time-window constraints. *Computers & Industrial Engineering*, *120*, 382–391.

Gao, Z., & Zhou, B. (2019). Bi-objective cyclic scheduling for single hoist with processing-time-window constraints considering buffer-setting. *J. Systems & Control Engineering*, *233*.

Gasperoni, F., & Schwiegelshohn, U. (1994). Generating close to optimum loop schedules on parallel processors. *Parallel Processing Letters*, *4*, 391–403.

Gultekin, H., Akturk, M. S., & Karasan, O. E. (2006). Cyclic scheduling of a 2-machine robotic cell with tooling constraints. *European Journal of Operational Research*, *174*, 777 – 796.

Hanen, C. (1994). Study of a NP-hard cyclic scheduling problem: The recurrent job-shop. *European Journal of Operational Research*, *72*, 82 – 101.

Hanen, C. (2009). Cyclic scheduling. In Y. Robert, & F. Vivien (Eds.), *Introduction to Scheduling* chapter 5. Springer.

Hanzalek, Z. (1998). A parallel algorithm for gradient training of feedforward neural networks. *Parallel Computing*, *24*, 823–839.

Hanzalek, Z., & Hanen, C. (2015). The impact of core precedences in a cyclic RCPSP with precedence delays. *Journal of Scheduling*, *18*, 275–284.

Hanzalek, Z., & Jurcik, P. (2010). Energy efficient scheduling for cluster-tree wireless sensor networks with time-bounded data flows: Application to IEEE 802.15.4/ZigBee. *Industrial Informatics, IEEE Transactions on*, *6*, 438 –450.

IEEE (2006). *IEEE 802.15.4 wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)*. IEEE SA Standards Board.

Kats, V., & Levner, E. (2011a). Cyclic routing algorithms in graphs: Performance analysis and applications to robot scheduling. *Computers & Industrial Engineering*, *61*, 279–288.

Kats, V., & Levner, E. (2011b). A faster algorithm for 2-cyclic robotic scheduling with a fixed robot route and interval processing times. *European Journal of Operational Research*, *209*, 51–56.

Kats, V., & Levner, E. (2018). On the existence of dominating 6-cyclic schedules in four-machine robotic cells. *European Journal of Operational Research*, *268*, 755–759.

Khatib, J., Kordon, A. M., Klikpo, E. C., & Trabelsi-Colibet, K. (2016). Computing latency of a real-time system modeled by synchronous dataflow graph. In *Proceedings of the 24th International Conference on Real-Time Networks and Systems, RTNS 2016, Brest, France, October 19-21, 2016* (pp. 87–96).

Levner, E., & Kats, V. (1998). A parametric critical path problem and an application for cyclic scheduling. *Discrete Applied Mathematics*, *87*, 149–158.

Levner, E., Kats, V., & de Pablo, D. A. L. (2007). Cyclic scheduling in robotic cells: An extension of basic models in machine scheduling theory. In E. Levner (Ed.), *Multiprocessor Scheduling: Theory and Applications* (pp. 1–20). Vienna Austria: I-Tech Education and Publishing.

Marchetti, O., & Munier-Kordon, A. (2009). Cyclic scheduling for the synthesis of embedded systems. In Y. Robert, & F. Vivien (Eds.), *Introduction to Scheduling* chapter 6. Springer.

Munier, A. (1996). The complexity of a cyclic scheduling problem with identical machines and precedence constraints. *European Journal of Operational Research*, *91*, 471 – 480.

Palopoli, L., Passerone, R., & Rizano, T. (2011). Scalable offline optimization of industrial wireless sensor networks. *Industrial Informatics, IEEE Transactions on*, *7*, 328–339.

Pempera, J., & Smutnicki, C. (2018). Open shop cyclic scheduling. *European Journal of Operational Research*, *269*, 773–781.

Potts, C. N., & Kovalyov, M. Y. (2000). Scheduling with batching: A review. *European Journal of Operational Research*, *120*, 228 – 249.

Proth, J. M., & Xie, X. (1995). *Modélisation, analyse et optimisation des systèmes à fonctionnement cyclique*. Masson.

ZigBee (2006). *ZigBee Specification, No. 053474r13*. ZigBee Standards Org.