Evaluation of the Age Latency of a Real-Time Communicating System Using the LET Paradigm

Alix Munier Kordon
Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
Alix.Munier@lip6.fr

Ning Tang
Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
Ning.Tang@lip6.fr

Abstract

Automotive and avionics embedded systems are usually composed of several tasks that are subject to complex timing constraints. In this context, the LET paradigm was introduced to improve the determinism of a system of tasks that communicate data through shared variables. The age latency corresponds to the maximum time for the propagation of data in these systems. Its precise evaluation is an important and challenging question for the design of these systems.

We consider in this paper a set of multi-periodic tasks that communicate data following the LET paradigm. Our main contribution is the development of mathematical and algorithmic tools to model precisely the dependency between tasks executions to experiment with an original methodology for computing the age latency of the system. These tools allow to handle the whole graph instead of particular chains and to extract automatically the critical parts of the graph. Experiments on randomly generated graphs indicate that systems with up to 90 periodic tasks and a hyperperiod bounded by 100 can be handled within a reasonable amount of time.

2012 ACM Subject Classification Computer systems organization → Real-time systems

Keywords and phrases Real-Time Systems, Logical Execution Time, Age Latency

Digital Object Identifier 10.4230/LIPIcs.ECRTS.2020.20

1 Introduction

A real-time system is a system that responds in a timely fashion to external events created by its environment [18]. In various contexts such as avionics or automotive, these systems must verify hard timing constraints. Their design and analysis are usually complex processes that require efficient methods.

We consider in this paper a set $\mathcal{T}$ of periodic tasks with different periods that are executed following the model of Liu and Layland [19]. A directed acyclic graph $\mathcal{G} = (\mathcal{T}, E)$ defines communication links between task executions. Each arc $(t_i, t_j) \in E$ between the two tasks t_i and t_j is associated to a shared memory variable that is modified by t_i and read by t_j. We assume that each execution of t_i updates the variable at its completion time, while each execution of t_j reads it at its starting time. This communication scheme, usually known as "implicit communication" follows the AUTOSAR requirements [1] and is commonly used for the design of automotive real-time systems.

However, the instants of the exchanges between tasks depend on the successive starting and completion times of the tasks, and are thus not predictable. The Logical Execution Time (LET) paradigm [15] delays writes to the periodic deadlines of the tasks and advances reads to their periodic release dates. The communication instants are then fixed before the execution of the tasks and the system is deterministic. This communication scheme was implemented by the time-triggered language Giotto [12]. This timing predictability makes it particularly suitable for safety-critical applications. This model was thus considered in

32nd Euromicro Conference on Real-Time Systems (ECRTS 2020).
Editor: Marcus Völp; Article No. 20; pp. 20:1–20:20
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

industrial domains like automotive [4, 10] and avionics [13, 23]. We suppose in this paper that tasks are periodic with different periods and that all communications follow the LET paradigm.

A real-time system usually communicates with its environment through sensors that detect events and actuators that transduce its reactions. Paths from a sensor to an actuator are usually referred to as event chains (see, for example, [10]). The time needed to propagate data from a sensor to an actuator is closely related to the reaction delay of the system. Several measures can be defined to capture these delays, as presented by Feiertag et al. [8]. We limit our study to the age latency, also called the end-to-end latency, which is the maximum time interval from a specific input value on a sensor to the last corresponding output value. It can be interpreted as the maximum delay that a specific data element spends in the system. This value measures the freshness of data producing a response of the system, and ensures that the actions of actuators are not too old.

The main contribution of the paper is to develop a general framework to model communications on successive task executions using LET communications for a general task dependency graph. The computation of the age latency of the application can then be seen as an example of a concrete application. This value cannot be defined in the presence of cycles in the dependency graph, thus graphs are assumed to be cycle-free. However, the transformations presented in this paper can be considered for general graphs. Observe that most of authors limit their methods to a single event chain [2, 8, 20].

Indeed, we first prove that dependencies induced by a LET communication $e = (t_i, t_j) \in E$ between the successive executions of t_i and t_j can be modelled by an original simple inequality involving parameters of the tasks t_i and t_j and the execution numbers considered.

Then, it can be observed that, if T_i denotes the period of task t_i, these dependency relations between task executions are repeated within the hyperperiod $T = lcm_{t_i \in \mathcal{T}}(T_i)$. An expanded valued graph $P_N(\mathcal{G})$ can then be built by duplicating each task $N_i = \frac{T}{T_i}$ times. We prove in this paper that setting any vector K with $K_i \in \mathbb{N} - \{0\}$ for any $t_i \in \mathcal{T}$, a partial expanded graph $P_K(\mathcal{G})$ can be built by duplicating each task K_i times. Each arc of this graph includes the modelling of the dependency relation between the corresponding executions of its adjacent task duplicates. This partial expanded graph is inspired from Bodin et al. [5] and de Groote [7] for Synchronous DataFlow Graphs [17], for which the initial inequality modelling dependency is slightly different.

Subsequently, we show that upper bounds on the latency between adjacent duplicates of $P_K(\mathcal{G})$ can be derived and considered as a valuation of the arcs. The longest paths of $P_K(\mathcal{G})$ then provide an upper bound on the latency. However, the computation of these paths has a time complexity proportional to $\sum_{e=(t_i,t_j)\in E} K_i \times K_j$. The main problem is then to find the value of K that minimises this function with an exact evaluation of the age latency.

We first prove that our study can be limited to vectors K such that, for any task t_i, K_i divides N_i. We then develop a greedy algorithm that converges to a vector $K^\star$ that provides the exact value of the age latency. This algorithm can be seen as an adaptation of the K-iter algorithm [6] for the determination of the maximum throughput of a Synchonous DataFlow Graph, which is up to now one of the best algorithms to solve this latter problem. Our algorithm was experimentally tested on randomly generated graphs with periods inspired from automotive real-life benchmarks [11, 16].

Our paper is organised as follows. Section 2 presents related work. The problem and our characterisation of the dependencies between tasks executions are presented in Section 3. Section 4 is devoted to the construction of the partial expanded graph $P_K(\mathcal{G})$ for any fixed vector K. It is shown in Section 5 that exploration can be limited to K vectors such that, for any task $t_i \in \mathcal{T}$, K_i is a divisor of N_i. Section 6 presents our greedy algorithm for the computation of a vector $K^\star$ leading to the exact value of the age latency. In section 7, we experiment with this algorithm on the ROSACE case study. Section 8 presents experiments on randomly generated graphs. Section 9 is our conclusion.

2 Related work

The evaluation of the age latency of an event chain is a challenging question tackled by several authors. Feiertag et al. [8] first introduced the definition of dependency between tasks of an event chain and four metrics to evaluate the delay between a sensor and an actuator. Becker et al. [2] developed a general framework to evaluate the age latency of an event chain using feasible intervals. They built an expanded graph by evaluating the possible propagation of input data by the successive executions of tasks. They tested in [3] their approach against the evaluation of the latency of a fixed schedule or under the LET hypothesis. They concluded that if there is no information on the communications or on the schedule, a pessimistic value of the age latency will be obtained, which is very similar to the value obtained using the LET paradigm. However, the computation time grows exponentially with the number of tasks if an enumeration is needed, while it remains constant for the LET paradigm.

Under the LET assumption, the times of the communications between tasks are known before the executions of the tasks. This strong assumption allows to characterise the dependencies between tasks if their parameters are fixed. Martinez et al. [20] gave a formal characterisation of the dependencies between tasks in an event chain using time instants. They then derived the age latency by enumerating all the possible paths of the corresponding expanded graph. They also proved that the release times influence the age latency and they developed a heuristic to fix them in order to minimise it.

Many practical applications are composed of graphs with no particular assumption on their structure [16, 22]. None of these previous approaches can be easily extended to these graphs. Indeed, the number of paths between two vertices is potentially exponential. The complexity of a method that enumerates all the paths for evaluating their age latency will thus grow exponentially following the parameters of the graph. Anyway, mainly two frameworks referenced below are capable of tackling such applications.

Pagetti et al. [21] have developed a language to express the constraints and a multi-periodic synchronous model to represent the whole system for a general graph. The size of the description of the communications is then equivalent to the one of the expanded graph $P_N(\mathcal{G})$. Forget et al. [9] showed that this approach supports several metrics.

Khatib et al. [14] proved that constraints between the successive executions of two adjacent tasks can be modelled using a Synchronous DataFlow Graph [17]. Our equation is slightly different since for any arc $e = (t_i, t_j)$, they did not not consider the successive constraints between two adjacent tasks if $T_i > T_j$, dealing only with precedence constraints. They then computed the age latency using the expansion of the Synchronous DataFlow Graph which is equivalent to $P_N(\mathcal{G})$. They also proposed the computation of a polynomial upper bound on the age latency equivalent to the determination of the longest paths of $P_{\mathbb{1}^n}(\mathcal{G})$ with $n = |\mathcal{T}|$. Lastly, they showed that the difference between this bound and the age latency is on average between 10 and 15 percent. This result motivates the development of efficient methods to evaluate more precisely the age latency of a graph $\mathcal{G}$.

3 Modelling of the system

This section formally presents the problem tackled in this paper. Subsection 3.1 defines the periodic tasks model considered according to LET restrictions. Subsection 3.2 is dedicated to the definition of the dependency relation between the successive executions of two adjacent tasks. Subsection 3.3 formally defines the age latency of a graph. Subsection 3.4 is devoted to the definition of the problem and the presentation of a small pedagogical example.

3.1 Periodic tasks model considering LET communications

Let us consider a set $\mathcal{T} = \{t_1, \ldots, t_n\}$ of real-time periodic tasks following the model of Liu and Layland [19]. Each task $t_i \in \mathcal{T}$ is characterised by a quadruple (r_i, C_i, D_i, T_i) such that:

- r_i is the release date (the offset) of the first execution of t_i;
- C_i is the worst-case execution time of t_i;
- D_i is the relative deadline of t_i;
- T_i is the period of t_i.

For any value $n \in \mathbb{N} - \{0\}$, we denote by $\langle t_i, n\rangle$ the nth execution of t_i and by $s(t_i, n)$ its starting time. The execution of $\langle t_i, n\rangle$ must be scheduled in its time window, that is $r_i + (n-1) \times T_i \leq s(t_i, n)$ and $s(t_i, n) + C_i \leq D_i + (n-1) \times T_i$.

The LET communication model separates task executions from communications. In this model, data are read at the release dates of reading tasks and written at the deadlines of writing tasks. Moreover, reading tasks always get the last emitted data. The main advantage of this model is to define a deterministic communications system even if tasks are delayed inside their time windows.

In this paper, we only consider LET communications and we limit the characterization of tasks to their successive time windows. The execution time associated to the nth execution of t_i is then set to its release date, that is, $\mathcal{S}(t_i, n) = r_i + (n-1) \times T_i$. Similarly, the completion time is fixed to $\mathcal{S}(t_i, n) + D_i$. Each task t_i is then given by the triple (r_i, D_i, T_i).

3.2 LET dependencies

Communications are expressed by a directed graph $\mathcal{G} = (\mathcal{T}, E)$. Each arc $e = (t_i, t_j) \in E$ induces dependencies between executions of t_i and t_j, defined as follows:

▶ **Definition 1.** *Let us suppose that $e = (t_i, t_j) \in E$ and that ν_i and ν_j are two positive integers. There exists a dependency relation from $\langle t_i, \nu_i\rangle$ to $\langle t_j, \nu_j\rangle$ if $\langle t_j, \nu_j\rangle$ receives data from $\langle t_i, \nu_i\rangle$ that is if:*

1. *The execution time of $\langle t_j, \nu_j\rangle$ is greater than or equal to the completion time of $\langle t_i, \nu_i\rangle$ and*
2. *the execution time of $\langle t_i, \nu_i + 1\rangle$ is greater than the completion time of $\langle t_j, \nu_j\rangle$ (since the data element from $\langle t_i, \nu_i + 1\rangle$ is not available for $\langle t_j, \nu_j\rangle$).*

Figure 1 presents successive time windows of the first executions of two periodic tasks t_1 and t_2 with a LET communication $e = (t_1, t_2) \in E$. Since $T_1 > T_2$ a single write from t_1 can be read by several executions of t_2. As an example, there is a dependency from $\langle t_1, 2\rangle$ to $\langle t_2, 4\rangle$ since $\langle t_1, 2\rangle$ ends before the beginning of $\langle t_2, 4\rangle$ and the data written by $\langle t_1, 3\rangle$ is not available at the beginning of $\langle t_2, 4\rangle$.

The next theorem characterises the dependency relation between the executions of two communicating tasks using the parameters of the executions:

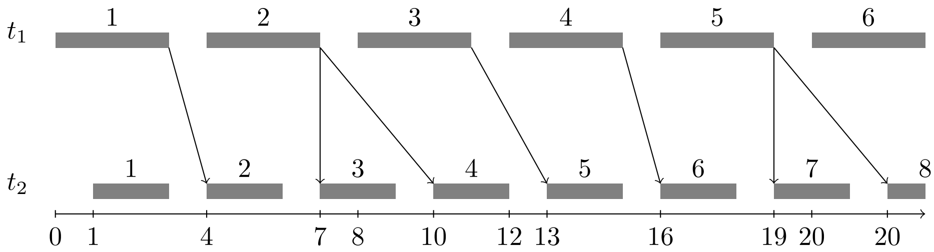


Figure 1 Time windows associated to two periodic tasks t_1 and t_2 with a LET dependency $e = (t_1, t_2)$. Parameters of tasks are respectively $(r_1, D_1, T_1) = (0, 3, 4)$ and $(r_2, D_2, T_2) = (1, 2, 3)$.

▶ **Theorem 2.** *Let $e = (t_i, t_j) \in E$, $gcd^e_T = \gcd(T_i, T_j)$ and the delay of e, $M^e = T_j + \left\lceil \frac{r_i - r_j + D_i}{gcd^e_T} \right\rceil \times gcd^e_T$. For any pair $(\nu_i, \nu_j) \in \mathbb{N} - \{0\} \times \mathbb{N} - \{0\}$, there exists a dependency from $\langle t_i, \nu_i \rangle$ to $\langle t_j, \nu_j \rangle$ iff $T_i \geq M^e + T_i \nu_i - T_j \nu_j > 0$.*

Proof. Following Definition 1, there exists a dependency from $\langle t_i, \nu_i \rangle$ to $\langle t_j, \nu_j \rangle$ if:

1. $\langle t_j, \nu_j \rangle$ begins after the completion of $\langle t_i, \nu_i \rangle$, thus $\mathcal{S}(t_i, \nu_i) + D_i \leq \mathcal{S}(t_j, \nu_j)$. Since $\mathcal{S}(t_i, \nu_i) = r_i + (\nu_i - 1) \times T_i$ and $\mathcal{S}(t_j, \nu_j) = r_j + (\nu_j - 1) \times T_j$, we get

$$r_i + (\nu_i - 1) \times T_i + D_i \leq r_j + (\nu_j - 1) \times T_j,$$

thus,

$$T_i \geq T_j + (r_i - r_j + D_i) + T_i \nu_i - T_j \nu_j,$$

and since in the inequality above only $r_i - r_j + D_i$ cannot be divided by gcd^e_T, we obtain that $T_i \geq M^e + T_i \nu_i - T_j \nu_j$.

2. The completion time of $\langle t_i, \nu_i + 1 \rangle$ is strictly greater than the execution time of $\langle t_j, \nu_j \rangle$, thus $\mathcal{S}(t_i, \nu_i + 1) + D_i > \mathcal{S}(t_j, \nu_j)$ and then

$$r_i + \nu_i T_i + D_i > r_j + (\nu_j - 1) \times T_j,$$

thus,

$$T_j + (r_i - r_j + D_i) + T_i \nu_i - T_j \nu_j > 0.$$

Since $M^e \geq T_j + (r_i - r_j + D_i)$, $M^e + T_i \nu_i - T_j \nu_j > 0$.

Merging the two inequalities gives the theorem. ◀

Let us consider, for example, the two tasks t_1 and t_2 with the LET communication $e = (t_1, t_2)$ presented in Figure 1. We get $\gcd^e_T = \gcd(3, 4) = 1$ and $M^e = 3 + (0 - 1 + 3) = 5$. The inequality of Theorem 2 is $4 \geq 5 + 4\nu_1 - 3\nu_2 \geq 0$. One can observe that the first executions of t_1 and t_2 with a dependency relation correspond to the pairs that verify this inequality. For $(\nu_1, \nu_2) = (1, 2)$, we get $5 + 4\nu_1 - 3\nu_2 = 5 + 4 - 6 = 3 \in \{1, \ldots, 4\}$. Similarly, for $(\nu_1, \nu_2) = (2, 3)$, we get $5 + 4\nu_1 - 3\nu_2 = 5 + 8 - 9 = 4 \in \{1, \ldots, 4\}$. Now, if we consider $(\nu_1, \nu_2) = (2, 5)$, $5 + 4\nu_1 - 3\nu_2 = 5 + 8 - 15 = -2 \notin \{1, \ldots, 4\}$ and there is no dependency from $\langle t_1, 2 \rangle$ to $\langle t_2, 5 \rangle$.

3.3 Age latency

Let us suppose that $e = (t_i, t_j) \in E$ and let $\mathcal{R}(e)$ be the set of pairs $(\nu_i, \nu_j) \in (\mathbb{N} - \{0\})^2$ such that e induces a dependency from $\langle t_i, \nu_i \rangle$ to $\langle t_j, \nu_j \rangle$. Then, for any pair $(\nu_i, \nu_j) \in \mathcal{R}(e)$, we define the latency of e between the executions $\langle t_i, \nu_i \rangle$ and $\langle t_j, \nu_j \rangle$ as

$$\mathcal{L}_{\nu_i,\nu_j}(e) = \mathcal{S}(t_j, \nu_j) - \mathcal{S}(t_i, \nu_i) = r_j - r_i + T_i - T_j - (T_i\nu_i - T_j\nu_j). \tag{1}$$

Now, for any path $p = t_1 t_2 \dots t_k$ of $\mathcal{G}$, we set $e_\ell = (t_\ell, t_{\ell+1})$ for the corresponding arcs with $\ell \in \{1, \dots, k-1\}$. We define $\mathcal{R}(p)$ as the set of k-tuples $(\nu_1, \dots, \nu_k) \in (\mathbb{N} - \{0\})^k$ such that $\forall \ell \in \{1, \dots, k-1\}$, $(\nu_\ell, \nu_{\ell+1}) \in \mathcal{R}(e_\ell)$. Then, for any k-tuple $(\nu_1, \dots, \nu_k) \in \mathcal{R}(p)$, we have

$$\mathcal{L}_{\nu_1,\dots,\nu_k}(p) = \sum_{\ell=1}^{k-1} \mathcal{L}_{\nu_\ell,\nu_{\ell+1}}(e_\ell) + D_k.$$

The age latency of a path p of $\mathcal{G}$ is then defined as the maximum time interval from a specific input value $\langle t_1, \nu_1 \rangle$ to the end of the output value $\langle t_1, \nu_1 \rangle$, thus

$$\mathcal{L}^\star(p) = \max\{\mathcal{L}_{\nu_1,\dots,\nu_k}(p), (\nu_1, \dots, \nu_k) \in \mathcal{R}(p)\}$$

and the maximum latency of a directed graph $\mathcal{G}$ corresponds to

$$\mathcal{L}^\star(\mathcal{G}) = \max\{\mathcal{L}^\star(p), p \text{ path of } \mathcal{G}\}.$$

Let us observe that, if the initial graph $\mathcal{G}$ contains cycles, its latency may not be bounded. Indeed, infinite paths p can be built in this case by looping in the cycles and the latency cannot be defined. So, we suppose in the following that $\mathcal{G}$ is acyclic. Moreover, since the latency between two executions is positive, $\mathcal{L}^\star(\mathcal{G})$ is reached for a path p such that t_1 has no predecessor and t_k no successor.

If $\mathcal{G}$ contains cycles, other definitions of the latency could be considered as "last-to-first" or "first-to-first", following Feiertag et al.'s definition [8]. The methodology and the algorithms presented in this paper can clearly be extended to tackle these cases and the existence of cycles does not complicate most of the reasoning.

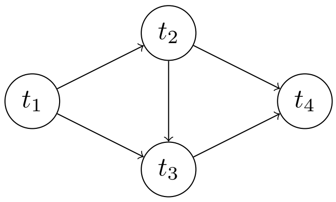
3.4 Problem definition and example

The problem tackled in this paper can be formalised as follows: let us consider a directed acyclic graph $\mathcal{G} = (\mathcal{T}, E)$, each arc modelling a LET communication. Each periodic task $t_i \in \mathcal{T}$ is associated to a triple (r_i, D_i, T_i). The problem is to compute the maximal age latency $\mathcal{L}^\star(\mathcal{G})$.

Figure 2 presents an instance of our problem comprising four periodic tasks and the associated directed acyclic graph $\mathcal{G}$. Dependency relations between the first executions of tasks t_1, t_2 and t_4 are shown in Figure 3, following the path $p = t_1 t_2 t_4$ of $\mathcal{G}$. The latency of the path from $\langle t_1, 1 \rangle$ to $\langle t_4, 1 \rangle$ is $\mathcal{L}_{1,2,1}(p) = \mathcal{S}(t_4, 1) - \mathcal{S}(t_1, 1) + 3 = 3 - 0 + 3 = 6$. In the same way, the latency of the path p from $\langle t_1, 3 \rangle$ to $\langle t_4, 2 \rangle$ is $\mathcal{L}_{3,5,2}(p) = \mathcal{S}(t_4, 2) - \mathcal{S}(t_1, 3) + 3 = 6 - 4 + 3 = 5$. We deduce that $\mathcal{L}^\star(p) = 6$.

4 Construction of a partial expanded graph

The aim of this section is to detail and prove the construction of a partial expanded graph $P_K(\mathcal{G})$ associated to a fixed vector $K \in (\mathbb{N} - \{0\})^n$. The main idea is to duplicate each task t_i, K_i times and to express the dependencies directly on duplicates.



t_i	t_1	t_2	t_3	t_4
r_i	0	1	2	3
D_i	1	0.5	4	3
T_i	2	1	6	3

Figure 2 An instance of four periodic tasks and the associated DAG $\mathcal{G}$.

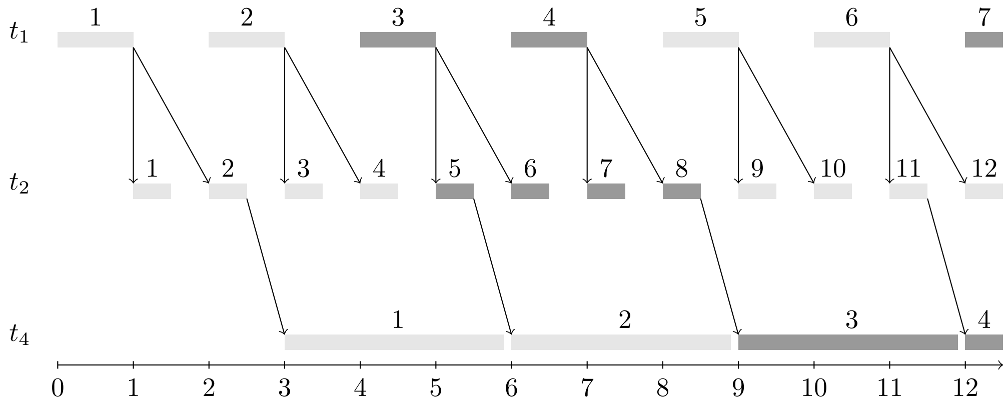


Figure 3 A path $p = t_1t_2t_4$ from the graph $\mathcal{G}$ shown in Figure 2. Time windows are colored following blocks of $K_1 = 2$ executions of t_1, $K_2 = 4$ executions of t_2 and $K_4 = 2$ executions of t_4.

Subsection 4.1 is devoted to the proof of Theorem 5 that characterises the dependency relations between the duplicates of two adjacent tasks. An upper bound on the latency between two duplicates corresponding to dependant executions is then evaluated in Subsection 4.2. Subsection 4.3 formally defines the partial expanded graph $P_K(\mathcal{G})$ associated with a vector K, while subsection 4.4 evaluates the complexity of its computation.

4.1 Characterisation of the dependencies between duplicates of the partial expanded graph

Let us suppose that for any task t_i, a number of duplicates $K_i \in \mathbb{N} - \{0\}$ is fixed. Then, for any $a_i \in \{1, \ldots, K_i\}$, the a_ith duplicate of t_i is simply associated to the executions $a_i + pK_i$ for $p \in \mathbb{N}$. For example, let us suppose that the task t_2 has a fixed number of duplicates $K_2 = 4$. For any value $a_2 \in \{1, 2, 3, 4\}$, we merge into a unique duplicate all the executions $\langle t_2, a_2 + pK_2\rangle$ for $p \in \mathbb{N}$. For $a_2 = 1$, it corresponds to executions $\langle t_2, 1\rangle, \langle t_2, 5\rangle, \langle t_2, 9\rangle \ldots \langle t_2, 1 + 4p\rangle$.

Now, suppose that $K_2 = 4$, $K_4 = 2$. We aim to characterize the dependencies from duplicates of t_2 to duplicates of t_4 due to the LET communication $e = (t_2, t_4)$. We observe in Figure 3 that there exists a dependency from $\langle t_2, 11\rangle$ to $\langle t_4, 4\rangle$. Moreover, $11 = 3 + 2 \times 4$ and $4 = 2 + 1 \times 2$. So, we set $a_2 = 3$, $a_4 = 2$ and we look to characterize dependencies from executions $\nu_2 = a_2 + p_2K_2 = 3 + 4p_2$ of t_2 to executions $\nu_4 = a_4 + p_4K_4 = 2 + 2p_4$ of t_4.

Following Theorem 2, the delay associated to e is $M^e = 3 + \left\lceil \frac{1-3+0.5}{1} \right\rceil = 2$. Moreover, there exists a dependency from $\langle t_2, \nu_2\rangle$ to $\langle t_4, \nu_4\rangle$ if and only if $T_2 \geq M^e + T_2\nu_2 - T_4\nu_4 > 0$.

Now, with these previous assumptions, $T_2\nu_2 - T_4\nu_4 = (3+4p_2) - 3(2+2p_4) = (4p_2 - 6p_4) - 3$. This difference is composed by a linear function of p_2 and p_4 and a constant term equal to 3.

These two terms are characterized in next lemma. Moreover, since $T_2 = 1$ we observe that, $M^e + T_2\nu_2 - T_4 = (4p_2 - 6p_4) - 1 = 1$, and thus $4p_2 - 6p_4 = 2$.

The conclusion is that there exists a dependency from $\langle t_2, 3 + 4p_2\rangle$ to $\langle t_4, 2 + 2p_4\rangle$ if and only if $4p_2 - 6p_4 = 2$. Theorem 5 generalizes this characterization to any LET communication between two communicating tasks.

▶ **Lemma 3.** *Consider $e = (t_i, t_j) \in E$ and let gcd_T^e (resp., gcd_K^e) be the greatest common divisor between T_i and T_j (resp., K_iT_i and K_jT_j). Let $\nu_i = a_i + p_iK_i$ and $\nu_j = a_j + p_jK_j$ with $(a_i, a_j) \in \{1, \ldots, K_i\} \times \{1, \ldots, K_j\}$ and $(p_i, p_j) \in \mathbb{N} \times \mathbb{N}$. Let us define the four values*

- $\alpha_e(a_i, a_j) = \dfrac{T_ia_i - T_ja_j}{gcd_T^e}$,
- $\pi_e(p_i, p_j) = \dfrac{T_ip_iK_i - T_jp_jK_j}{gcd_K^e}$,
- $\pi_e^{max}(a_i, a_j) = \left\lfloor \dfrac{-M^e + T_i - \alpha_e(a_i, a_j) \cdot gcd_T^e}{gcd_K^e} \right\rfloor$ *and*
- $\pi_e^{min}(a_i, a_j) = \left\lceil \dfrac{-M^e + gcd_T^e - \alpha_e(a_i, a_j) gcd_T^e}{gcd_K^e} \right\rceil$.

If e induces a dependency from $\langle t_i, \nu_i\rangle$ to $\langle t_j, \nu_j\rangle$, then

$$T_i\nu_i - T_j\nu_j = \pi_e(p_i, p_j) \cdot gcd_K^e + \alpha(a_i, a_j) \cdot gcd_T^e$$

with $\pi_e(p_i, p_j) \in \{\pi_e^{min}(a_i, a_j), \ldots, \pi_e^{max}(a_i, a_j)\}$.

Proof. By definition of ν_i and ν_j,

$$\begin{aligned} T_i\nu_i - T_j\nu_j &= T_i \times (a_i + K_ip_i) - T_j \times (a_j + K_jp_j) = (T_iK_ip_i - T_jK_jp_j) + (T_ia_i - T_ja_j) \\ &= \pi_e(p_i, p_j) \cdot gcd_K^e + \alpha_e(a_i, a_j) \cdot gcd_T^e. \end{aligned}$$

By Theorem 2, $T_i - M^e \geq T_i\,\nu_i - T_j\nu_j > -M^e$. Thus, since all the terms of this inequality are divisible by gcd_T^e, it is equivalent to $T_i - M^e \geq T_i\,\nu_i - T_j\nu_j \geq -M^e + gcd_T^e$ and we get

$$T_i - M^e \geq \pi_e(p_i, p_j) \cdot gcd_K^e + \alpha_e(a_i, a_j) \cdot gcd_T^e \geq -M^e + gcd_T^e.$$

From the right part of the inequality,

$$\pi_e(p_i, p_j) \geq \frac{-M^e + gcd_T^e - \alpha_e(a_i, a_j) \cdot gcd_T^e}{gcd_K^e}.$$

Since $\pi_e(p_i, p_j)$ is an integer, we can tighten the lower bound of $\pi_e(p_i, p_j)$ by

$$\pi_e(p_i, p_j) \geq \left\lceil \frac{-M^e + gcd_T^e - \alpha_e(a_i, a_j) \cdot gcd_T^e}{gcd_K^e} \right\rceil = \pi_e^{min}(a_i, a_j).$$

In the same way, the left part of the previous inequality is

$$\frac{T_i - M^e - \alpha_e(a_i, a_j) \cdot gcd_T^e}{gcd_K^e} \geq \pi_e(p_i, p_j).$$

Since $\pi_e(p_i, p_j)$ is an integer, we can tighten the upper bound on $\pi_e(p_i, p_j)$ by

$$\left\lfloor \frac{T_i - M^e - \alpha_e(a_i, a_j) \cdot gcd_T^e}{gcd_K^e} \right\rfloor \geq \pi_e(p_i, p_j)$$

So we get $\pi_e^{max}(a_i, a_j) \geq \pi_e(p_i, p_j)$ and the lemma is proved. ◀

Table 1 Values $\alpha_e(a_2, a_4)$, $\pi_e^{max}(a_2, a_4)$ and $\pi_e^{min}(a_2, a_4)$ for $a_2 \in \{1, 2, 3, 4\}$ and $a_4 \in \{1, 2\}$.

$a_2 \backslash a_4$	1	2
1	−2	−5
2	−1	−4
3	0	−3
4	1	−2

$\alpha_e(a_2, a_4)$

$a_2 \backslash a_4$	1	2
1	0	2
2	0	1
3	−1	1
4	−1	0

$\pi_e^{max}(a_2, a_4)$

$a_2 \backslash a_4$	1	2
1	1	2
2	0	2
3	0	1
4	−1	1

$\pi_e^{min}(a_2, a_4)$

Consider as an example, the arc $e = (t_2, t_4)$ of the example shown in Figure 2 with fixed values $K_2 = 4$ and $K_4 = 2$. We get $gcd_T^e = \gcd(1, 3) = 1$, $gcd_K^e = \gcd(4, 6) = 2$ and $M^e = 2$. The corresponding values of $\alpha_e(a_i, a_j)$, $\pi_e^{max}(a_i, a_j)$ and $\pi_e^{min}(a_i, a_j)$ are shown in Table 1.

For the pair $(a_2, a_4) = (3, 2)$, suppose that there exists a dependency from $\langle t_2, \nu_2\rangle$ to $\langle t_4, \nu_4\rangle$ with $\nu_2 = a_2 + p_2K_2 = 3 + 4p_2$ and $\nu_4 = a_4 + p_4K_4 = 2 + 2p_4$.

$$T_2\nu_2 - T_4\nu_4 = \nu_2 - 3\nu_4 = (3+4p_2) - 3(2+2p_4) = 2(2p_2 - 3p_4) - 3 = gcd_K^e \cdot \pi_e(p_2, p_4) - \alpha_e(3, 2).$$

As $\pi_e^{max}(3, 2) = \pi_e^{min}(3, 2) = 1$, the only possible value for $\pi_e(p_2, p_4)$ is 1, thus $\pi_e(p_2, p_4) = 2p_2 - 3p_4 = 1$.

Consider now the pair $(a_2, a_4) = (1, 1)$. Then, since $\pi_e^{max}(1, 1) < \pi_e^{min}(1, 1)$, such a decomposition of the difference $T_2\nu_2 - T_4\nu_4$ with $\nu_2 = 1 + p_2K_2$ and $\nu_4 = 1 + p_4K_4$ is not possible; a simple consequence of Lemma 3 is that there is no dependency relation between executions $\langle t_2, 1 + p_2K_2\rangle$ and $\langle t_4, 1 + p_4K_4\rangle$.

We observe in Figure 3 that there exist dependencies $\langle t_2, 2\rangle \to \langle t_4, 1\rangle$, $\langle t_2, 5\rangle \to \langle t_4, 2\rangle$, $\langle t_2, 8\rangle \to \langle t_4, 3\rangle$ and $\langle t_2, 11\rangle \to \langle t_4, 4\rangle$. They correspond respectively to the pairs $(a_2, a_4) = (2, 1)$, $(a_2, a_4) = (1, 2)$, $(a_2, a_4) = (4, 1)$ and $(a_2, a_4) = (3, 2)$. For all these pairs, one can check that $\pi_e^{max}(a_2, a_4) \geq \pi_e^{min}(a_2, a_4)$.

For the general case, a consequence of Lemma 3 is that there is no dependency between executions $\langle t_i, a_i + p_iK_i\rangle$ and $\langle t_j, a_j + p_jK_j\rangle$ if $\pi_e^{max}(a_i, a_j) < \pi_e^{min}(a_i, a_j)$. Thus, let us define

$$\mathbb{A}(e) = \left\{(a_i, a_j) \in \{1, \ldots, K_i\} \times \{1, \ldots, K_j\} \mid \pi_e^{max}(a_i, a_j) \geq \pi_e^{min}(a_i, a_j)\right\}.$$

For our particular case, $\mathbb{A}(e) = \{(2, 1), (1, 2), (4, 1), (3, 2)\}$.

The next lemma is the converse of Lemma 3.

▶ **Lemma 4.** *Let $e = (t_i, t_j) \in E$ and $(a_i, a_j) \in \mathbb{A}(e)$. For any integer value $\pi \in \{\pi_e^{min}(a_i, a_j), \ldots, \pi_e^{max}(a_i, a_j)\}$, there exists an infinite number of pairs $(p_i, p_j) \in \mathbb{N}^2$ such that $\pi = \pi_e(p_i, p_j)$. Moreover, setting $\nu_i = a_i + p_iK_i$ and $\nu_j = a_j + p_jK_j$, e induces a dependency from $\langle t_i, \nu_i\rangle$ to $\langle t_j, \nu_j\rangle$.*

Proof. By Bezout's identity, there exists $(x, y) \in \mathbb{Z}^2$ such that $xK_iT_i + yK_jT_j = gcd_K^e$ and thus $\pi xK_iT_i + \pi yK_jT_j = \pi \cdot gcd_K^e$.

For $z \in \mathbb{N}$, let us define $p_i = \pi x + zK_jT_j$ and $p_j = -\pi y + zK_iT_i$. Let us also consider values ν_i and ν_j such that $\nu_i = a_i + K_ip_i$ and $\nu_j = a_j + K_jp_j$. For z sufficiently large ($z \geq z_0$), $p_i \geq 1$ and $p_j \geq 1$, and thus ν_i and ν_j are both greater than 1. Then,

$$\begin{aligned} T_ip_iK_i - T_jp_jK_j &= K_iT_i(\pi x + zK_jT_j) - K_jT_j(-\pi y + zK_iT_i) \\ &= x\pi K_iT_i + y\pi K_jT_j = \pi \cdot gcd_K^e, \end{aligned}$$

thus $\pi = \pi_e(p_i, p_j)$. Now,

$$T_i\nu_i - T_j\nu_j = a_iT_i - a_jT_j + K_iT_ip_i - K_jZ_jp_j = a_iT_i - a_jT_j + \pi \cdot gcd_K^e$$

and thus, by definition of α_e, $T_i\nu_i - T_j\nu_j = \alpha_e(a_i, a_j) \cdot gcd_T^e + \pi \cdot gcd_K^e$. Recall now that $\pi \in \{\pi_e^{min}(a_i, a_j), \ldots, \pi_e^{max}(a_i, a_j)\}$, thus

$$T_i\nu_i - T_j\nu_j \leq \alpha_e(a_i, a_j) \cdot gcd_T^e + \pi_e^{max}(a_i, a_j) \cdot gcd_K^e,$$

and, since $\pi_e^{max}(a_i, a_j) \cdot gcd_K^e \leq -M^e + T_i - \alpha_e(a_i, a_j) \cdot gcd_T^e$,

$$T_i\nu_i - T_j\nu_j \leq -M^e + T_i. \tag{2}$$

Similarly, since $\pi_e^{min}(a_i, a_j) \cdot gcd_K^e \geq -M^e + gcd_T^e - \alpha_e(a_i, a_j) \cdot gcd_T^e$,

$$\begin{aligned} T_i\nu_i - T_j\nu_j &\geq \pi_e^{min}(a_i, a_j)gcd_K^e + \alpha_e(a_i, a_j)gcd_T^e \\ &\geq -M^e + gcd_T^e > -M^e. \end{aligned} \tag{3}$$

From equations (2) and (3), we have $T_i \geq M^e + T_i\nu_i - T_j\nu_j > 0$ and by Theorem 2 there is a dependency from $\langle t_i, \nu_i\rangle$ to $\langle t_j, \nu_j\rangle$. The lemma is proved. ◀

From Lemmas 3 and 4, we deduce the following main theorem:

▶ **Theorem 5.** *Let t_i and t_j be two tasks such that t_i (resp.t_j) is duplicated K_i (resp.K_j) times. Let $e = (t_i, t_j) \in E$ and $(a_i, a_j) \in \{1, \ldots, K_i\} \times \{1, \ldots, K_j\}$. There exists a dependency relation from $\langle t_i, a_i + p_iK_i\rangle$ to $\langle t_j, a_j + p_jK_j\rangle$ for $(p_i, p_j) \in \mathbb{N}^2$ iff $\pi_e^{min}(a_i, a_j) \leq \pi_e(p_i, p_j) \leq \pi_e^{max}(a_i, a_j)$.*

4.2 Upper bound on the latency

For any arc $e = (t_i, t_j) \in E$ and any pair $(a_i, a_j) \in \mathbb{A}(e)$, Theorem 5 gives the existence of a dependency from some executions $\langle t_i, \nu_i\rangle$ to $\langle t_j, \nu_j\rangle$ with $\nu_i = a_i + p_iK_i$ and $\nu_j = a_j + p_jK_i$. In order to evaluate the age latency of the whole graph $\mathcal{G}$, the next theorem evaluates the maximum latency associated to these executions of t_i and t_j.

▶ **Theorem 6** (Upper bound on the latency between two tasks)**.** *Let t_i and t_j be two tasks such that t_i (resp.t_j) is duplicated K_i (resp.K_j) times. Let also $e = (t_i, t_j) \in E$ and $(a_i, a_j) \in \mathbb{A}(e)$. Then*

$$\mathcal{L}_{(a_i,a_j)}^{max}(e) = r_j - r_i + T_i - T_j - (\pi_e^{min}(a_i, a_j) \cdot gcd_K^e + \alpha_e(a_i, a_j) \cdot gcd_T^e)$$

is the maximal value of the latency $\mathcal{L}_{\nu_i,\nu_j}(e)$ for $(\nu_i, \nu_j) \in \mathcal{R}(e)$ with $\nu_i = a_i \bmod K_i$ and $\nu_j = a_j \bmod K_j$.

Proof. By Equation (1), the latency between executions $\langle t_i, \nu_i\rangle$ and $\langle t_j, \nu_j\rangle$ for $(\nu_i, \nu_j) \in \mathcal{R}(e)$ is $\mathcal{L}_{\nu_i,\nu_j}(e) = r_j - r_i + T_i - T_j - (T_i\nu_i - T_j\nu_j)$. Assuming that $\nu_i = a_i + p_iK_i$ and $\nu_j = a_j + p_jK_j$ with $(p_i, p_j) \in \mathbb{N}^2$ we have by Lemma 3 that

$$\mathcal{L}_{\nu_i,\nu_j}(e) = r_j - r_i + T_i - T_j - (\pi_e(p_i, p_j) \cdot gcd_K^b + \alpha_b(a_i, a_j) \cdot gcd_T^b) \tag{4}$$

By Theorem 5, $\pi_e(p_i, p_j) \in \{\pi_e^{min}(a_i, a_j), \ldots, \pi_e^{max}(a_i, a_j)\}$. We conclude that $\mathcal{L}_{\nu_i,\nu_j}(e)$ is maximum for $\pi_e(p_i, p_j) = \pi_e^{min}(a_i, a_j)$ and the theorem is proved. ◀

4.3 Definition of the partial expanded graph

We suppose that the vector $K \in (\mathbb{N} - \{0\})^n$ is fixed. The associated expanded graph $P_K(\mathcal{G}) = (V, B, \mathcal{L}^{max})$ is a valued directed acyclic graph defined as follows:

1. Each task t_i is duplicated K_i times. For any value $a \in \{1, \ldots, K_i\}$, the ath duplicate of t_i is denoted by t_i^a and is associated to the executions $\langle t_i, a + pK_i \rangle$ for $p \in \mathbb{N}$.
2. For any arc $e = (t_i, t_j) \in E$, we build an arc (t_i^a, t_j^b) for every pair $(a, b) \in \{1, \ldots, K_i\} \times \{1, \ldots, K_j\}$ if $\pi_e^{max}(a, b) \geq \pi_e^{min}(a, b)$.
3. For every arc $\beta = (t_i^a, t_j^b) \in B$, $\mathcal{L}^{max}(\beta) = \mathcal{L}^{max}_{(a,b)}(e)$ following Theorem 6.
4. Lastly, two additional fictitious tasks s and f are considered with the arcs defined as:
   - For any duplicate t_i^a with no predecessors, add the arc $\beta = (s, t_i^a)$ with $\mathcal{L}^{max}(\beta) = 0$;
   - For any duplicate t_i^a with no successors, add the arc $\beta = (t_i^a, f)$ with $\mathcal{L}^{max}(\beta) = D_i$.

Let us denote by $LP^{max}(P_K(\mathcal{G}))$ the length of the longest path of the associated partial expanded graph $P_K(\mathcal{G})$ considering the arcs values $\mathcal{L}^{max}(\beta)$, $\beta \in B$. By Theorem 6, values on the arcs of $P_K(\mathcal{G})$ are upper bounds of the age latency, thus $LP^{max}(P_K(\mathcal{G}))$ is an upper bound of the maximum latency of $\mathcal{G}$.

Figure 4 presents the expanded graph $P_K(\mathcal{G})$ associated with the vector $K = (2, 4, 1, 2)$ for the instance shown in Figure 2. A longest path is given by $p = s, t_1^1, t_3^1, t_4^1, f$ with a corresponding length equal to 12, i.e., $LP^{max}(P_K(\mathcal{G})) = 12$. We conclude that $\mathcal{L}^\star(\mathcal{G}) \leq LP^{max}(P_K(\mathcal{G})) = 12$.

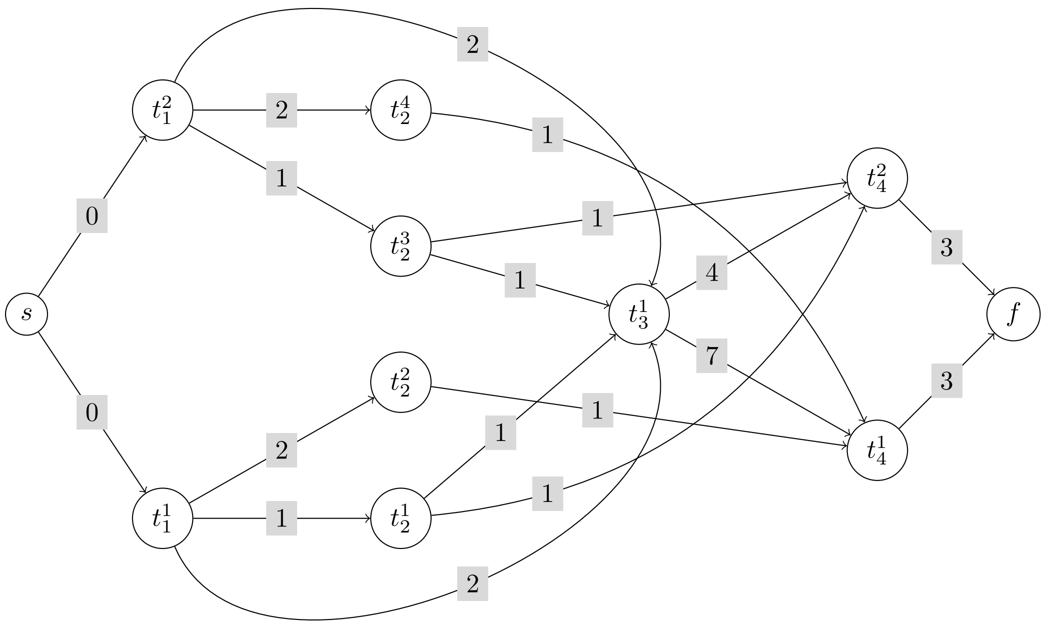


Figure 4 Expanded graph $P_K(\mathcal{G}) = (V, B, \mathcal{L}^{max})$ for the instance shown in Figure 2 associated with the vector $K = (2, 4, 1, 2)$. Arcs $\beta \in B$ are weighted by $\mathcal{L}^{max}(\beta)$ in gray.

4.4 Complexity of the computation of $P_K(\mathcal{G})$ and its longest paths

$P_K(\mathcal{G})$ is a graph without cycles. Thus, the computation of the longest paths can be done in time complexity $\Theta(|V| + |B|)$ by simply sorting the vertices following a topological order used in the next step to explore the vertices.

Note that the total number of vertices of $P_K(\mathcal{G})$ is $|V| = \sum_{i=1}^{n} K_i + 2$, while the number of arcs $|B|$ is bounded by $\mathcal{O}(\sum_{e=(t_i,t_j)\in E} K_i \times K_j)$. These two values may be huge for large values of K. The main problem consists then in the determination of the vector K of small values such that the bound $LP^{max}(P_K(\mathcal{G}))$ is as close as possible to the age latency $\mathcal{L}^\star(\mathcal{G})$.

5 Dominant set for the expansion vector K

This section is devoted to the study of dominance properties on K w.r.t the age latency to reduce the set of vectors K. In Subsection 5.1 we prove that the value of the longest paths of the expanded graph $P_N(\mathcal{G})$ associated with the hyperperiod N of $\mathcal{G}$ is the age latency $\mathcal{L}^\star(\mathcal{G})$. We prove in Subsection 5.2 that we can reduce our study to the set of the partial expansions $P_K(\mathcal{G})$ such that each component K_i divises N_i and we provide a partial order relation between these vectors that will be exploited in the following section for the computation of the age latency of $\mathcal{G}$.

5.1 Maximal value of the age latency for $K = N$

Consider $T = lcm_{t_i \in \mathcal{T}}(T_i)$ and the repetition vector $N \in \mathbb{N}^{*n}$ defined as $N_i = \dfrac{T}{T_i}$ for any task $t_i \in \mathcal{T}$. For our example shown in Figure 2, we get $T = lcm(2, 1, 6, 3) = 6$ and thus $N = (3, 6, 1, 2)$. Lemma 7 is a simple technical lemma.

▶ **Lemma 7.** *Let $P_N(\mathcal{G}) = (V, B, \mathcal{L}^{max})$ be the expanded graph with $K = N$, $e = (t_i, t_j)$ be an arc of $\mathcal{G}$. For any arc $\beta = (t_i^{a_i}, t_j^{a_j}) \in B$ associated with e and any pair $(q_i, q_j) \in \mathbb{N}^2$, $\pi_e(q_i, q_j) = q_i - q_j$.*

Proof. By definition of π_e, $\pi_e(q_i, q_j) = \dfrac{T_i q_i K_i - T_j q_j K_j}{gcd_K^e}$. As $T_i K_i = T_j K_j = T = gcd_K^e$, we have $\pi_e(q_i, q_j) = q_i - q_j$ and the lemma is proved. ◀

We prove formally in the following that the value of the longest path of the expanded graph $P_N(\mathcal{G})$ is the age latency of $\mathcal{G}$, i.e., $\mathcal{L}^\star(\mathcal{G})$:

▶ **Theorem 8.** *For any acyclic directed graph $\mathcal{G}$, $LP^{max}(P_N(\mathcal{G})) = \mathcal{L}^\star(\mathcal{G})$.*

Proof. By Theorem 6 and the definition of the partial expanded graphs, $LP^{max}(P_N(\mathcal{G})) \geq \mathcal{L}^\star(\mathcal{G})$. We prove that $LP^{max}(P_N(\mathcal{G})) \leq \mathcal{L}^\star(\mathcal{G})$.

Consider a path $p_N = t_1^{a_1}, t_2^{a_2} \dots t_k^{a_k}$ of $P_N(\mathcal{G})$ and the corresponding path $p = t_1, t_2 \dots t_k$ of $\mathcal{G}$. We also set $e_\ell = (t_\ell, t_{\ell+1})$ for $\ell \in \{1, \dots, k-1\}$. By Lemma 7, we have for any vector $(q_1, \dots, q_k) \in \mathbb{N}^k$ and $\ell \in \{1, \dots, k-1\}$, $\pi_{e_\ell}(q_\ell, q_{\ell+1}) = q_\ell - q_{\ell+1}$. Let us consider the sequence of integers $\tilde{q}_1, \dots, \tilde{q}_k$ defined as follows:

- $\tilde{q}_{\ell+1} = \tilde{q}_\ell + \pi_{e_\ell}^{max}(a_\ell, a_{\ell+1})$
- $\tilde{q}_1$ is fixed sufficiently large such that, $\forall \ell \in \{1, \dots, k\}$, $\tilde{q}_\ell \geq 0$.

This sequence satisfies $\forall \ell \in \{1, \dots, k-1\}$, $\pi_{e_\ell}(\tilde{q}_\ell, \tilde{q}_{\ell+1}) = \pi_{e_\ell}^{max}(a_\ell, a_{\ell+1})$, thus by Theorem 5, there is a dependency relation from $\langle t_\ell, a_\ell + \tilde{q}_\ell K_\ell \rangle$ to $\langle t_{\ell+1}, a_{\ell+1} + \tilde{q}_{\ell+1} K_{\ell+1} \rangle$. Moreover, by the definition of the sequence of arcs β_ℓ, $\mathcal{L}^{max}(\beta_\ell) = \mathcal{L}_{\tilde{q}_\ell, \tilde{q}_{\ell+1}}(e_\ell)$ and then $\mathcal{L}_{\tilde{q}_1, \dots, \tilde{q}_k}(p) = LP^{max}(p_N)$. If p_N is the longest path $P_N(\mathcal{G})$, $LP^{max}(P_N(\mathcal{G})) = LP^{max}(p_N) = \mathcal{L}_{\tilde{q}_1, \dots, \tilde{q}_k}(p) \leq \mathcal{L}^\star(\mathcal{G})$, which proves the theorem. ◀

5.2 Order relation between the divisors of the repetition vector N

The next theorem introduces an order relation between vectors $K \in (\mathbb{N} - \{0\})^n$.

▶ **Theorem 9.** *For any acyclic directed graph $\mathcal{G}$, suppose that K and K' are two different vectors such that $\forall t_i \in \mathcal{T}$, K'_i is a divisor of K_i, then $LP^{max}(P_{K'}(\mathcal{G})) \geq LP^{max}(P_K(\mathcal{G}))$.*

Proof. Let us consider the arc $e = (t_i, t_j)$ of $\mathcal{G}$. By the hypothesis, there exists $(x_i, x_j) \in (\mathbb{N} - \{0\})^2$, such that $K_i = x_i K'_i$ and $K_j = x_j K'_j$. Let $\beta = (t_i^{a_i}, t_j^{a_j})$ be an arc of $P_K(\mathcal{G})$ with $(a_i, a_j) \in \{1, \dots, K_i\} \times \{1, \dots, K_j\}$. Then, following Theorem 6 and the definition of the partial expanded graph, there exists $(\nu_i, \nu_j) \in (\mathbb{N} - \{0\})^2$ such that $\nu_i = a_i + p_i K_i$, $\nu_j = a_j + p_j K_j$ and $\mathcal{L}_{\nu_i,\nu_j}(t_i, t_j) = \mathcal{L}^{max}(\beta)$.

Let us consider now integer values $a'_i \in \{1, 2, \dots, K'_i\}$, $a'_j \in \{1, 2, \dots, K'_j\}$, y_i and y_j such that $a_i = a'_i + y_i K'_i$ and $a_j = a'_j + y_j K'_j$. Thus, $\nu_i = a'_i + (y_i + x_i p_i) K'_i$ and $\nu_j = a'_j + (y_j + x_j p_j) K'_j$. Since there is a dependency relation between $\langle t_i, \nu_i \rangle$ and $\langle t_j, \nu_j \rangle$, $\beta' = (t_i^{a'_i}, t_j^{a'_j})$ belongs to $P_{K'}(G)$ and $\mathcal{L}_{\nu_i,\nu_j}(t_i, t_j) \leq \mathcal{L}^{max}(\beta')$, thus we get $\mathcal{L}^{max}(\beta) \leq \mathcal{L}^{max}(\beta')$.

For any path $p = t_1^{a_1}, t_2^{a_2}, \dots t_q^{a_q}$ in $P_K(\mathcal{G})$, there is a corresponding path $p' = t_1^{a'_1}, t_2^{a'_2}, \dots t_q^{a'_q}$ in $P_{K'}(G)$ that includes all executions represented by path p. Therefore, $LP^{max}(P_{K'}(\mathcal{G})) \geq LP^{max}(P_K(\mathcal{G}))$. ◀

For any pair of vectors $(K, K') \in (\mathbb{N} - \{0\})^n \times (\mathbb{N} - \{0\})^n$, we set $K' \preceq K$ if, for any $t_i \in \mathcal{T}$, K'_i divides K_i. By Theorem 8, the exact value of the latency is reached for $K = N$. The consequence of this last theorem is that we can limit our study to the set $\mathcal{K}$ of vectors $K \preceq N$. Let us consider the graph $H = (\mathcal{K}, \preceq)$. The evaluation of the age latency is improved following paths from $K = \mathbb{1}^n$ to $K = N$. A vector $K \in \mathcal{K}$ is said to be optimum if $LP^{max}(P_K(\mathcal{G})) = \mathcal{L}^\star(\mathcal{G})$.

Figure 5 shows the graph H associated with the example from Figure 2. We observe that the exact value $\mathcal{L}^\star(\mathcal{G})$ of the age latency can be reached for vectors K smaller than N, i.e., there are several optimum vectors. The next section presents an algorithm to compute an optimum vector.

6 Determination of an optimum vector $K^\star$

The problem considered in this section is to compute an optimum vector $K^\star$, i.e., such that $LP^{max}(P_{K^\star}(\mathcal{G})) = \mathcal{L}^\star(\mathcal{G})$. Our algorithm computes iteratively a vector $K \in \mathcal{K}$ until the optimality test expressed by the next lemma is true.

▶ **Lemma 10** (Optimality test)**.** *Consider a vector $K \in \mathcal{K}$, a longest path p_K of $P_K(\mathcal{G})$ and its corresponding path p of $\mathcal{G}$. If, for every task $t_i \in p$, K_i is a multiple of $N_i(p) = \dfrac{lcm_{t_j \in p}\{T_j\}}{T_i}$, then $LP^{max}(p_K) = \mathcal{L}^\star(\mathcal{G})$.*

Proof. Consider a vector K and the path p of $\mathcal{G}$ following the assumptions of the theorem. By definition of p_K, $LP^{max}(P_K(\mathcal{G})) = LP^{max}(p_K)$. We first prove that $\mathcal{L}^\star(p) = LP^{max}(p_K)$.

- Since p is a path of $\mathcal{G}$, $\mathcal{L}^\star(\mathcal{G}) \geq \mathcal{L}^\star(p)$. Now, by Theorem 6, $LP^{max}(P_K(\mathcal{G})) \geq \mathcal{L}^\star(\mathcal{G})$ and by definition of p_K, $LP^{max}(p_K) = LP^{max}(P_K(\mathcal{G}))$, thus $\mathcal{L}^\star(p) \leq LP^{max}(p_K)$.
- Now, since for any task t_i of p, $N_i(p)$ is a divisor of K_i, we have by Theorem 9 that $LP^{max}(P_{N(p)}(p)) \geq LP^{max}(p_K)$. Moreover, by Theorem 8, $LP^{max}(P_{N(p)}(p)) = \mathcal{L}^\star(p)$, thus $\mathcal{L}^\star(p) \geq LP^{max}(p_K)$.

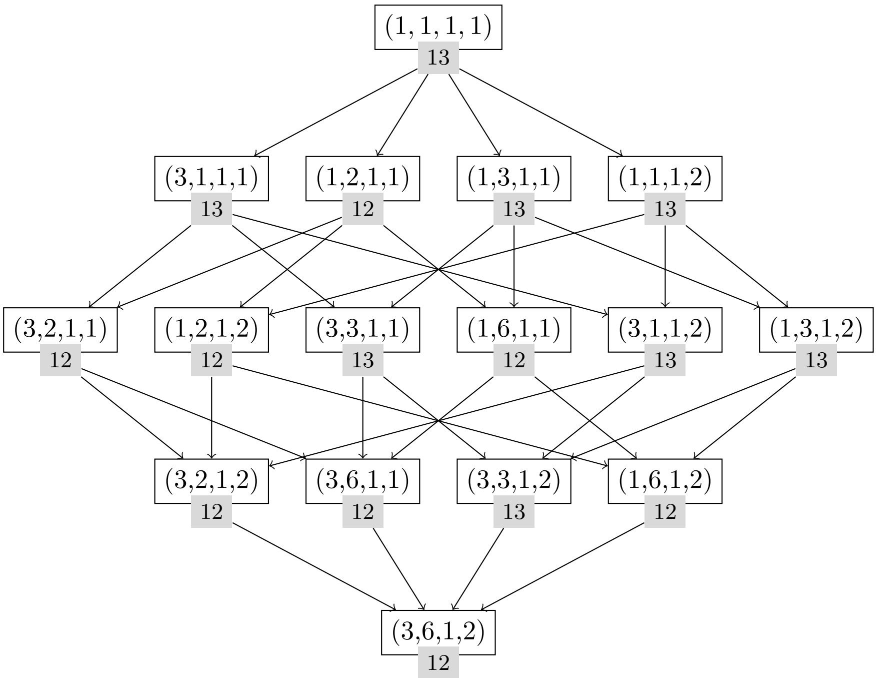


Figure 5 Graph $H = (\mathcal{K}, \preceq)$ associated with the example shown in Figure 2. Values $LP^{max}(P_K(\mathcal{G}))$ are given in gray for each vertex $K \in \mathcal{K}$.

So, we proved that $\mathcal{L}^\star(p) = LP^{max}(p_K) = LP^{max}(P_K(\mathcal{G}))$. Now, $\mathcal{L}^\star(\mathcal{G}) \geq \mathcal{L}^\star(p) = LP^{max}(p_K)$. Since $K \preceq N$, $\mathcal{L}^\star(\mathcal{G}) \leq LP^{max}(P_K(\mathcal{G})) = LP^{max}(p_K)$ by Theorem 9, and thus $LP^{max}(P_K(\mathcal{G})) = \mathcal{L}^\star(\mathcal{G}) = LP^{max}(p_K)$, which completes the proof. ◀

Algorithm 1 is inspired from the K-iter algorithm [6] which computes an expansion vector K for the determination of the optimum throughput of a Synchronous DataFlow Graph. For the initialisation phase, $K = \mathbb{1}^n$. K is simply increased at each step for tasks from the longest path of $P_K(\mathcal{G})$ until the maximality test is met.

Algorithm 1 Compute an optimum vector $K^\star$ and the age latency $\mathcal{L}(\mathcal{G})$.

Require: A DAG $\mathcal{G} = (\mathcal{T}, E)$, (r_i, D_i, T_i) for every $t_i \in \mathcal{T}$
Ensure: An optimum vector $K^\star$ and the age latency $\mathcal{L}^\star(\mathcal{G})$
  Set $K = \mathbb{1}^n$
  repeat
    Compute $P_K(\mathcal{G})$ and a longest path p_K of $P_K(\mathcal{G})$
    Set $p = s, t_1 \dots t_k, f$ to the corresponding path of $\mathcal{G}$
    Set $T(p) \leftarrow lcm(T_1, \dots, T_k)$ and $\forall i \in \{1, \dots, k\}$, $N_i(p) \leftarrow \frac{T(p)}{T_i}$
    OptPathFound$\leftarrow \forall t_i \in p, N_i(p) | K_i$
    if not OptPathFound **then**
      $\forall i \in \{1, \dots, k\}$, $K_i \leftarrow lcm(K_i, N_i(p))$
    end if
  until OptPathFound

Theorem 11 shows the convergence of the algorithm.

▶ **Theorem 11.** *For any directed acyclic graph $\mathcal{G}$, Algorithm 1 converges to a vector $K^\star \in \mathcal{K}$ such that $LP^{max}(P_{K^\star}(\mathcal{G})) = \mathcal{L}^\star(\mathcal{G})$.*

Proof. For any $q > 0$, we denote by $K(q)$ the vector K at the end of the qth iteration: $q = 0$ corresponds to the initialisation phase. We show that for any integer $q \geq 0$, $K(q) \in \mathcal{K}$ and $K(q) \preceq K(q+1)$ with $K(q) \neq K(q+1)$.

- At the initialisation step, $K(0) = \mathbb{1}^n \in \mathcal{K}$.
- Now, suppose that at step q, the optimality test is not true and that $K(q) \in \mathcal{K}$. Consider a task $t_i \in \mathcal{T}$. If t_i does not belong to p, $K_i(q+1) = K_i(q)$. Otherwise, $K_i(q+1) = lcm(K_i(q), N_i(p))$ where $K_i(q)$ and $N_i(p)$ are both divisors of N_i. Thus, $K_i(q+1)$ is also a divisor of N_i, and we get that $K(q+1) \in \mathcal{K}$ with $K(q) \preceq K(q+1)$.
- Lastly, we prove by contradiction that $K(q) \neq K(q+1)$. Indeed, suppose that $K_i(q) = K_i(q+1)$ for any task $t_i \in \mathcal{T}$, then since $K_i(q+1) = lcm(K_i(q), N_i(p))$, we deduce that $N_i(p)$ is a divisor of $K_i(q)$. Thus, the optimality test is true, which is a contradiction.

We conclude that vectors $K(q)$ are strictly increasing while the optimality test is false. By Lemma 10, the vector $K(q)$ is optimum when the optimality test is true. Lastly, the optimality test is true for the repetition vector N; this insures the convergence of the algorithm. ◀

The number of iterations of Algorithm 1 is not bounded and can be theoretically proportional to the maximum length of a path of the graph $H = (\mathcal{K}, E_{\preceq})$.

Let us consider the first step of Algorithm 1 for the example of Figure 2. At initialisation, $K = \mathbb{1}^4$. The corresponding partial expanded graph $P_K(\mathcal{G})$ is shown by Figure 6. Its longest path of $P_K(\mathcal{G})$ is $p_K = s, t_1^1, t_2^1, t_3^1, t_4^1, f$ valued by $LP^{max}(p_K) = 13$. The optimality test fails, and we get $N(p) = (3, 6, 1, 2)$ which is the repetition vector and thus $K^\star = K(1) = N$.

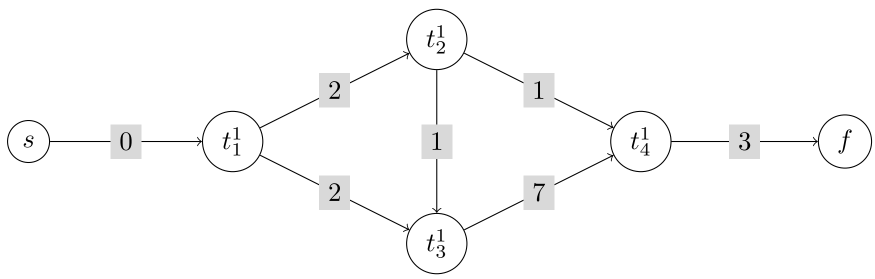


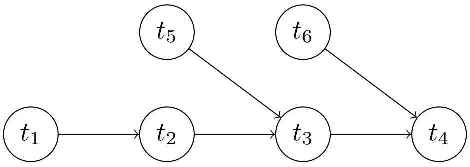
Figure 6 The partial expanded graph for the instance shown in Figure 2 and a unit vector $K = (1, 1, 1, 1)$. Arcs are weighted by $\mathcal{L}^{max}$ in gray.

7 ROSACE Case Study

ROSACE is the acronym for Research Open-Source Avionics and Control Engineering. This case study was developed by Pagetti et al. [22] to illustrate the implementation of a real-time system on a many-core architecture. Figure 7 presents an instance of the problem extracted from [9]. We arbitrarily set $r_i = 0$ and $D_i = T_i$ for any task $t_i \in \mathcal{T}$.

Figure 8 presents the partial expansion of the instance of Figure 7 for the unit expansion vector $K = \mathbb{1}^6$. A path of maximum length is $p_K = s, t_1^1, t_2^1, t_3^1, t_4^1, f$ with $LP^{max}(P_K(\mathcal{G})) = LP^{max}(p_K) = 260$ms.

At the first iteration of Algorithm 1, $p = s, t_1, t_2, t_3, t_4, f$ is expanded. We set $T(p) = \text{lcm}(60, 40, 30) = 120$, $N_1(p) = N_2(p) = 2$, $N_3(p) = 3$ and $N_4(p) = 4$. The next iteration, we set $K = (2, 2, 3, 4, 1, 1)$.



t_i	t_1	t_2	t_3	t_4	t_5	t_6
r_i	0	0	0	0	0	0
D_i	60	60	40	30	30	30
T_i	60	60	40	30	30	30

Figure 7 An instance of 6 periodic tasks and the associated DAG $\mathcal{G}$ extracted from the ROSACE case study [9].

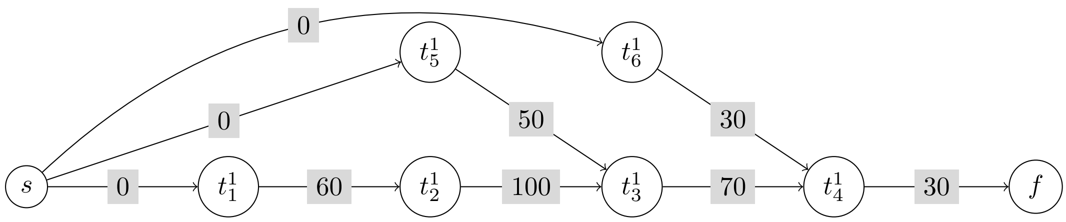


Figure 8 The partial expanded graph $P_K(\mathcal{G})$ for the instance shown in Figure 7 and a unit vector $K = \mathbb{1}^6$. Each arc β is weighted by $\mathcal{L}^{max}(\beta)$, shown in gray.

The partial expanded graph $P_K(\mathcal{G})$ built at the second iteration is shown in Figure 9. $p_K = s, t_1^1, t_2^2, t_3^2, t_4^4, f$ is a longest path of $P_K(\mathcal{G})$ with $LP^{max}(p_K) = LP^{max}(P_K(\mathcal{G})) =$ 240ms Moreover, the associated path $p = s, t_1, t_2, t_3, t_4, f$ verifies $T(p) = lcm(30, 40, 60)$, $N_1(p) = N_2(p) = 2$, $N_3(p) = 3$ and $N_4(p) = 4$. The optimality test is true and we get $K^\star = (2, 2, 3, 4, 1, 1)$. The maximum age latency of $\mathcal{G}$ is thus $\mathcal{L}^\star(\mathcal{G}) = LP^{max}(p_{K^\star}) = 240$ms.

We observe in this example that all the tasks of the critical path (i.e., the paths p of $\mathcal{G}$ such that $\mathcal{L}^\star(p) = \mathcal{L}^\star(\mathcal{G})$) were expanded at least following $N(p)$. Moreover, tasks from other paths are not necessarily duplicated: for example, $K_5^\star = K_6^\star = 1$ with $N_5 = N_6 = 4$. Thus, we can identify that paths s, t_5, t_3, t_4, f and s, t_6, t_4, f are not critical and tasks can be delayed without influence on the age latency.

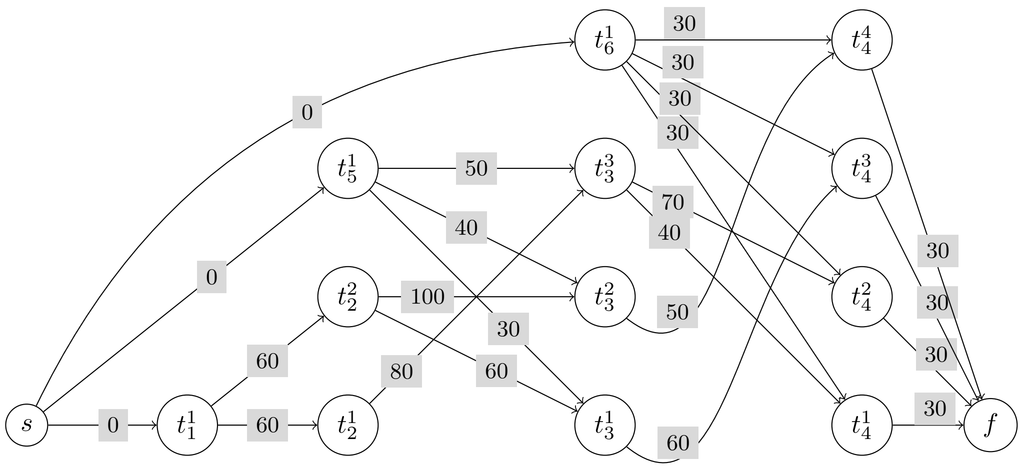


Figure 9 The partial expanded graph $P_K(\mathcal{G})$ for the instance shown in Figure 7 and the vector $K = (2, 2, 3, 4, 1, 1)$. Each arc β is weighted by $\mathcal{L}^{max}(\beta)$.

8 Experimental results

Our experiments aim at testing the performance of Algorithm 1. Following the experiments of Khatib et al. [14], the bound obtained from the longest paths of $P_{\mathbb{1}^n}(\mathcal{G})$ can be computed quickly, but its performance is on average between 10 and 15 percent from the maximal value $\mathcal{L}^\star(\mathcal{G})$. Moreover, their method does not precisely identify the real critical paths w.r.t the age latency of the initial graph.

Our Benchmarks were randomly generated: they are detailed is Subsection 8.1. The analysis of the computation time of our algorithm is presented in Subsection 8.2. Subsection 8.3 deals with the analysis of the critical vectors $K^\star$ obtained by our algorithm.

All our experiments were performed on an Intel(R) Core(TM) i5-8400 CPU (6 cores at 2.80GHz) and 15 GB of RAM. Our codes are written in Python. Functions dealing with graphs were implemented using the Python package NetworkX.

The goal is to experimentally analyse properties of Algorithm 1, like the number of iterations, space and time complexity. We used linear regression and curve fitting to map these properties to the size and density of initial graphs graphs.

8.1 Benchmarks

Random instances of n tasks were generated as follows. Periods of tasks are selected uniformly in $\mathcal{H} = \{1, 2, 5, 10, 20, 50, 100\}$. $\mathcal{H}$ is a subset of the values presented by Kramer et al. [16] for the 2015 WATERS challenge and several authors dealing with the age latency for automotive applications [10, 3].

Release times r_i are uniformly selected in $\{0, 1, 2, 3, 4, 5\}$, while we fix the relative deadline D_i equal to the period of the task, i.e., $D_i = T_i$ for any task $t_i \in \mathcal{T}$. Graphs are randomly generated using the Python NetworkX function `dense_gnm_random_graph`. Nodes are arbitrary numbered from 1 to n. A directed acyclic graph is then built by replacing each edge $e = \{i, j\}$ with $i < j$ by an arc $e = (i, j)$.

For any number n of tasks, we set the number of arcs to $m_\ell = \left\lfloor \frac{(n(n-1)}{4} \right\rfloor$ for *low density* graphs and $m_h = \left\lceil \frac{(n(n-1)}{3} \right\rceil$ for *high density*. We start with $n = 5$ tasks with a step of 5. For each data point, 150 random instances were generated and an average value of the functions considered are shown.

8.2 Analysis of the computation time of Algorithm 1

For sufficiently large n, the hyperperiod of an instance is exactly $T = lcm\{\alpha \in \mathcal{H}\} = 100$. The consequence is that the number of duplicates (*resp.*, the number of arcs) of the expanded graph $P_N(\mathcal{G})$ is bounded by $T \times n$ (*resp.*, $T^2 \times n^2$).

We measured the running time and the number of iterations of Algorithm 1. We stopped at $n = 90$ tasks, since the running time exceeded 15 minutes on average for instances with higher values of n. Figure 10 reports the average running times and Figure 11 the average number of iterations following the number of tasks.

We observed that the running time of Algorithm 1 is a quadratic function of the number of tasks, and thus is linear in the number of arcs of the graph $\mathcal{G}$. Unsurprisingly, these running times are longer for high-density graphs. This observation seems to contradict the experimental results of Becker et al. [3]: indeed, they remarked that the average running time for the computation of the age latency of a chain is linear w.r.t the number of tasks. In this case, the number of arcs equals $n - 1$: the running time is then also linear w.r.t the number of arcs, which is coherent with our result.

We also noticed that the whole number of iterations of Algorithm 1 grows logarithmically on average. Our first experimental conclusion is thus that the convergence of the algorithm to the exact value seems to be a logarithmic function of the number of tasks. The long running time is thus due to the time needed to build the successive partial expansions and not to the increase of the number of iterations of the algorithm.

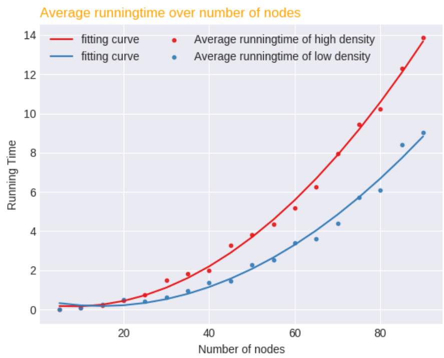


Figure 10 Average running times w.r.t the number of nodes. Fitting functions presented are $f_h(n) = (2.02\times10^{-3})n^2 - 0.03n + 0.29$ and $f_\ell(n) = (1.53 \times 10^{-3})n^2 - 0.05n + 0.51$ for respectively high-density and low-density graphs.

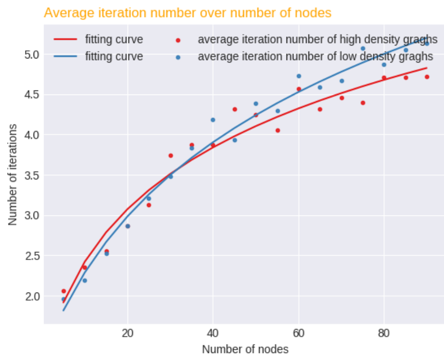


Figure 11 Average number of iterations w.r.t the number of nodes. Fitting functions presented are $g_h(n) = 1.34\ln(0.62(n+5.89)) - 0.64$ and $g_\ell(n) = 1.96\ln(1.59(n+13.42)) - 4.81$ for respectively high-density and low-density graphs.

8.3 Analysis of the partial expanded graph obtained

Figure 12 presents the evolution of the ratio $r(n) = \dfrac{\sum_{i=1}^{n} K_i^\star}{\sum_{i=1}^{n} N_i}$ following the number of tasks and the density of the graph. We observed that it is roughly a linear function that remains bounded by 0.8 for high-density graphs and 0.65 for low-density ones. The consequence is that in many cases we clearly do not need to completely expand the graph to get the exact value of the age latency and that good algorithms should be sought to identify the critical paths of a graph.

9 Conclusion

In this paper, we present a new definition of the dependency between the successive executions of two tasks that communicate following the LET paradigm. This definition was exploited to build a partial expanded graph $P_K(\mathcal{G})$ associated to any vector $K \in (\mathbb{N} - \{0\})^n$ for the computation of an upper bound of the age latency. A greedy algorithm to compute an accurate value $K^\star$ leading to the exact value of the age latency was developed and tested on random instances. This optimal partial expansion allows to identify the critical paths of the graph $\mathcal{G}$.

Many extensions of our study may be considered. The performance of our algorithm should be improved by building the successive partial expended graphs incrementally and optimizing data structures for graphs. Our methodology can surely be applied to evaluate accurate lower bounds of the age latency. Coupling the upper and the lower bounds will allow then to precisely measure the error between the longest paths of $P_K(\mathcal{G})$ and $\mathcal{L}^\star(\mathcal{G})$.

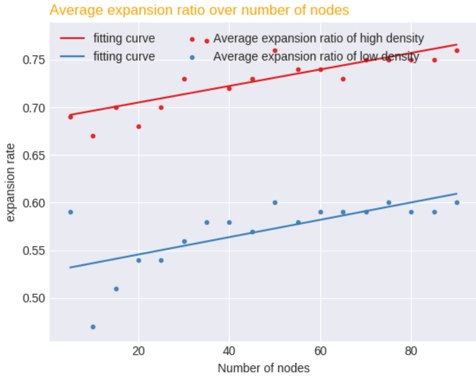


Figure 12 Average ratio $r(n) = \frac{\sum_{i=1}^{n} K_i^\star}{\sum_{i=1}^{n} N_i}$ for the partial expanded graph computed by Algorithm 1. Fitting functions presented are $r_h(n) = 8.67 \times 10^{-4}n + 0.69$ and $r_\ell(n) = 9.1 \times 10^{-4}n + 0.52$ for respectively high-density and low-density graphs.

Our general framework should also be extended to tackle other possible latencies [8]. Lastly, an implicit communication between two tasks of same period (which corresponds to two tasks in the same runnable for an AUTOSAR compatible system) could easily be considered in our model.

References

1 Autosar. URL: `https://www.autosar.org`.

2 Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, and Thomas Nolte. Synthesizing job-level dependencies for automotive multi-rate effect chains. In *2016 IEEE 22nd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 159–169, August 2016. `doi:10.1109/RTCSA.2016.41`.

3 Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, and Thomas Nolte. End-to-end timing analysis of cause-effect chains in automotive embedded systems. *Journal of Systems Architecture*, 80:104–113, 2017.

4 Alessandro Biondi and Marco Di Natale. Achieving predictable multicore execution of automotive applications using the LET paradigm. In *IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS 2018, 11-13 April 2018, Porto, Portugal*, pages 240–250, 2018. `doi:10.1109/RTAS.2018.00032`.

5 Bruno Bodin, Alix Munier Kordon, and Benoît Dupont de Dinechin. K-periodic schedules for evaluating the maximum throughput of a synchronous dataflow graph. In *2012 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, SAMOS XII, Samos, Greece, July 16-19, 2012*, pages 152–159, 2012.

6 Bruno Bodin, Alix Munier Kordon, and Benoît Dupont de Dinechin. 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*, pages 160:1–160:6, 2016.

7 Robert de Groote. *On the analysis of synchronous dataflow graphs: a system-theoretic perspective*. PhD thesis, University of Twente, 2016.

8 Nico Feiertag, Kai Richter, Johan Nordlander, and Jan Jonsson. A compositional framework for end-to-end path delay calculation of automotive systems under different path semantics. In *IEEE Real-Time Systems Symposium, November 30-December 3*. IEEE Communications Society, 2009.

9 Julien Forget, Frédéric Boniol, and Claire Pagetti. Verifying end-to-end real-time constraints on multi-periodic models. In *22nd IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2017, Limassol, Cyprus, September 12-15, 2017*, pages 1–8, 2017.

10 Arne Hamann, Dakshina Dasari, Simon Kramer, Michael Pressler, and Falk Wurst. Communication centric design in complex automotive embedded systems. In *29th Euromicro Conference on Real-Time Systems, ECRTS 2017, June 27-30, 2017, Dubrovnik, Croatia*, pages 10:1–10:20, 2017.

11 Arne Hamann, Dakshina Dasari, Simon Kramer, Michael Pressler, Falk Wurst, and Dirk Ziegenbein. Waters industrial challenge 2017. URL: `https://waters2017.inria.fr/challenge/#Challenge17`.

12 Thomas A. Henzinger, Benjamin Horowitz, and Christoph M. Kirsch. Giotto: a time-triggered language for embedded programming. *Proceedings of the IEEE*, 91(1):84–99, 2003.

13 Thomas A. Henzinger, Christoph M. Kirsch, Marco A.A Sanvido, and Wolfgang Pree. From control models to real-time code using Giotto. *IEEE Control Systems Magazine*, 23(1):50–64, February 2003.

14 Jad Khatib, Alix Munier Kordon, Enagnon Cédric Klikpo, and Kods Trabelsi-Colibet. 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*, pages 87–96, 2016.

15 Christoph M. Kirsch and Ana Sokolova. The logical execution time paradigm. In Samarjit Chakraborty and Jörg Eberspächer, editors, *Advances in Real-Time Systems*, pages 103–120. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

16 Simon Kramer, Dirk Ziegenbein, and Arne Hamann. Real world automotive benchmarks for free, 2015. URL: `https://www.ecrts.org/forum/viewtopic.php?f=20&t=23`.

17 Edward A. Lee and David G. Messerschmitt. Synchronous data flow. *Proceeding of the IEEE*, vol. 75(no. 9):pp. 1235–1245, 1987.

18 Qing Li and Caroline Yao. *Real-time concepts for embedded systems*. Taylor and Francis, Hoboken, NJ, 2014. URL: `http://cds.cern.ch/record/1990357`.

19 C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, 1973. `doi:10.1145/321738.321743`.

20 Jorge Martinez, Ignacio Sañudo, and Marko Bertogna. Analytical characterization of end-to-end communication delays with logical execution time. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2244–2254, November 2018.

21 Claire Pagetti, Julien Forget, Frédéric Boniol, Mikel Cordovilla, and David Lesens. Multi-task implementation of multi-periodic synchronous programs. *Discrete Event Dynamic Systems*, 21(3):307–338, 2011.

22 Claire Pagetti, David Saussié, Romain Gratia, Eric Noulard, and Pierre Siron. The ROSACE case study: from simulink specification to multi/many-core execution. In *2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 309–318, April 2014.

23 Rémy Wyss, Frédéric Boniol, Claire Pagetti, and Julien Forget. End-to-end latency computation in a multi-periodic design. In *28th Symposium On Applied Computing (SAC'13)*, pages 1682–1687, Coimbra, Portugal, April 2013.