Reliability and performance optimization of pipelined real-time systems

Anne Benoit*, Fanny Dufossé*, Alain Girault† Yves Robert*
* Laboratoire de l'Informatique du Parallélisme
École Normale Supérieure de Lyon, France
{Anne.Benoit, Fanny.Dufosse, Yves.Robert}@ens-lyon.fr
† INRIA Grenoble Rhône-Alpes, France
Alain.Girault@inria.fr

***Abstract*—We consider pipelined real-time systems, commonly found in assembly lines, consisting of a chain of tasks executing on a distributed platform. Their processing is pipelined: each processor executes only one interval of consecutive tasks. We are therefore interested in minimizing both the input-output latency and the period. For dependability reasons, we are also interested in maximizing the reliability of the system. We therefore assign several processors to each interval of tasks, so as to increase the reliability of the system. We assume that both processors and communication links are unreliable and subject to transient failures, the arrival of which follows a constant parameter Poisson law. We also assume that the failures are statistically independent events. We study several variants of this multiprocessor mapping problem with several hypotheses on the target platform (homogeneous/heterogeneous speeds and/or failure rates). We provide NP-hardness complexity results, and optimal mapping algorithms for polynomial problem instances.**

Keywords: *Pipelined real-time systems, interval mapping, multi-criteria (reliability, latency, period) optimization, complexity results.*

I. Introduction

Pipelined real-time systems are commonly found in assembly lines and are subject to strict *dependability* and *real-time constraints*. They consist of a chain of tasks executing on a distributed platform. Each task is a block of code with a known amount of work to be processed. The role of the first task of the chain is to acquire some data set from the environment (thanks to sensor drivers), to process it, and finally to transmit its result to the second task. Each subsequent task receives its input data from its immediately preceding task, processes it, and transmits its result to its immediately successor task, except the last task that transmits it to the environment (thanks to actuator drivers).

Tasks are assigned to processors of the platform using an *interval mapping*, which groups consecutive tasks of the linear chain and assigns them to the same processor. Interval mappings are more general than one-to-one mappings, which establish a unique correspondence between tasks and processors; they are very useful for reducing communication overheads, not to mention the many situations where there are more tasks than processors and where interval mappings are mandatory. The key performance-oriented metrics to determine the best interval mapping are the *period* and the *latency*. The period is the time interval between the beginning of the execution of two consecutive data sets. Equivalently, the inverse of the period is the *throughput*, which measures the aggregate rate of processing of data. The latency is the time elapsed between the beginning and the end of the execution of a given data set; hence, it measures the response time of the system for processing the data set entirely. Minimizing the latency is *antagonistic* to minimizing the period, and tradeoffs should be found between these two criteria.

Besides real-time constraints, expressed as an upper bound on the period and/or the latency, pipelined real-time systems must also satisfy crucial *dependability constraints*, which are expressed as a lower bound on the *reliability* of the mapping. Increasing the reliability is achieved by replicating the intervals on several processors. Augmenting the replication level (defined as the average number of times each interval is replicated) is good for the reliability, but bad for the period and latency, because less processors will be available for executing the intervals of tasks. We thus have three antagonistic criteria: reliability, period, and latency.

We evaluate the reliability of a single task mapped onto a processor according to the classical model of Shatz and Wang [21], where each hardware component (processor or communication link) is *fail-silent* and is characterized by a *constant failure rate per time unit* λ: the reliability of a task of duration d is therefore $e^{-\lambda d}$. For an interval of several tasks mapped onto a single processor, we just have to sum up the task durations, hence obtaining $e^{-\lambda D}$, where D is the sum of the interval's task durations. For a mapping with replication, we compute the reliability by building the *Reliability Block Diagram* (RBD) corresponding to this mapping. Here we face the delicate issue that computing the reliability is exponential in the size of the mapping (or equivalently the size of the RBD). To solve this issue, we insert *routing operations* in the mapping to guarantee that the RBD is by construction serial-parallel, therefore allowing us to compute its reliability in linear time.

We first present the models in Section II, and then discuss related work in Section III. The core of our contribution is presented in Sections IV, V, and VI. Finally, we conclude in Section VII.

II. Framework

In this section, we detail the application model, the platform model, the failure model, and the replication model. We end with the formal definition of the mono- and multi-criteria multiprocessor mapping problems.

A. Application model

An application is a *chain* of n tasks $\mathcal{C} = (\tau_i)_{1 \leq i \leq n}$. Each task τ_i is a block of code that (1) receives its input from its predecessor τ_{i-1}, (2) computes a known amount of work, (3) and produces an output data set of a known size. Therefore, each task τ_i is represented by the pair (w_i, o_i), where w_i is the amount of work and o_i is the output data size. By convention, $o_n = 0$ because τ_n emits its result directly to the environment through actuator drivers. Specifying the size of the input data set required by a task is not necessary since, by definition of a chain, it is equal to the size of the output data set of its immediately preceding task. Figure 1 shows an example of a chain composed of n tasks.

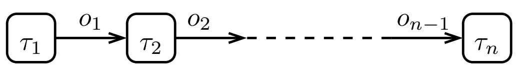

Figure 1. Example of a chain of n tasks.

Executing τ_i on a processor of speed s takes w_i/s units of time. Transmitting the result of τ_i on a link of bandwidth b takes o_i/b units of time. Knowing the values w_i and o_i is not a critical assumption since worst-case execution time (WCET) analysis has been applied with success to real-life processors actually used in embedded systems. In particular, it has been applied to the most critical existing embedded system, namely the Airbus A380 avionics software running on the Motorola MPC755 processor [9], [22].

B. Platform model

The target platform consists of p processors connected by point-to-point communication links. We note $\mathcal{P}$ the set of processors: $\mathcal{P} = (P_u)_{1 \leq u \leq p}$. We assume that communication links are *homogeneous*: this means that all links have the same bandwidth b. On the contrary, each processor P_u may have a different speed s_u. Such platforms correspond to networks of workstations with plain TCP/IP interconnects or other LANs.

In order to derive a realistic communication model, we assume that the number of outgoing point-to-point connections of each processor is limited to $\mathcal{K}$. A given processor is thus capable of simultaneously sending messages to (and receiving messages from) $\mathcal{K}$ other processors. Indeed, there is no physical device capable of sending, say, 100 messages to 100 distinct processors, at the same speed as if it was a single message. The output bandwidth of the sender's network card would be a limiting factor. Our assumption of bounded multi-port communications [14] is reasonable for a large range of platforms, from large-scale clusters to multi-core System-on-Chips (SoCs).

In addition, we assume that communications are *overlapped* with computations, that is, a processor can compute the current instance of task τ_i and, in parallel, send to another processor the result of the previous instance of τ_i. This model is consistent with current processor architectures where a SoC can include a main processor and several communication co-processors.

C. Interval mapping

The chain of tasks is executed repeatedly in a *pipelined manner* to achieve a better throughput. As a consequence, mapping the chain on the platform involves dividing the chain into m intervals of consecutive tasks, and assigning each processor to a unique interval. This technique is known as *interval mapping*. Figure 2 shows an example of a division of a chain of tasks into m intervals.

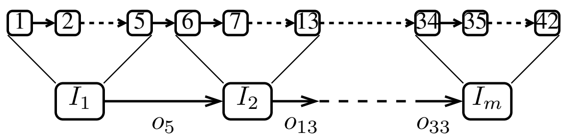

Figure 2. A chain of tasks divided into m intervals.

In a mapping without replication, each interval is assigned to a single processor. If the number of processors is greater than the number of tasks, then each interval can be of size one (that is, one task per interval), but this is rarely the case for real-life systems. Furthermore, having many small intervals is likely to decrease the period and the failure probability, but it will also increase the communication costs, and hence the latency: thus a trade-off is to be found.

In a mapping with replication, each interval is assigned to several processors. Replication is crucial to increase the reliability of the system [10].

For each $1 \leq j \leq m$, the interval I_j is the set of consecutive tasks between indices f_j and l_j. Moreover, $f_1 = 1$, $\forall 2 \leq j \leq m, f_j = l_{j-1} + 1$, and $l_m = n$. The amount of work processed by I_j is therefore $W_j = \sum_{\tau_i \in I_j} w_i = \sum_{i=f_j}^{l_j} w_i$. The size of the output data set produced by interval I_j is that of its last task, that is, o_{l_j}.

D. Failure model

Both processors and communication links can fail, and they are *fail-silent*. Classically, we adopt the failure model of Shatz and Wang [21]: failures are *transient* and the maximal duration of a failure is such that it affects only the current operation executing onto the faulty processor, and not the subsequent operations (same for communication links); this is the "hot" failure model. Besides, the occurrence of failures on a processor (same for a communication link) follows a Poisson law with a constant parameter λ, called its *failure rate per time unit*. Modern fail-silent hardware components can have a failure rate around 10^{-6} per hour.

Since communication links are homogeneous, we note λ_ℓ their identical failure rate per time unit. Concerning the processors, we note λ_u the failure rate per time unit of the processor P_u, for each P_u in $\mathcal{P}$.

Moreover, failure occurrences are *statistically independent events*. Note that transient failures are the most common failures in modern processors, all the more when processor voltage is lowered to reduce the energy consumption, because, in that case, even very low energy particles are likely to create a critical charge leading to a transient failure [25].

The *reliability* of a system measures its continuity of service. It is defined as the probability that it functions correctly during a given time interval [2]. According to our model, the reliability of the processor P (resp. the communication link L) during the duration d is $r = e^{-\lambda d}$, where λ is the failure rate per time unit of P or L. Conversely, the *probability of failure* of the processor P (resp. the communication link L) during the duration d is $f = 1 - r = 1 - e^{-\lambda d}$. Hence, the reliability of the task τ_i on processor P_u is:

$$r_{u,i} = e^{-\lambda_u \, w_i \, / \, s_u} \tag{1}$$

Accordingly, the reliability of the interval I mapped on the processor P_u is:

$$r_{u,I} = e^{-\lambda_u \, W_j \, / \, s_u} = \prod_{\tau_i \in I} r_{u,i} \tag{2}$$

Equations (1) and (2) show that platform heterogeneity may come from two factors: (i) processors having different speeds, and (ii) processors having different failure rates. We say that the platform is *homogeneous* if all the processors have the same speed s and the same failure rate λ (hence the reliability and the execution time of an interval no longer depends on the processor it is assigned to, and we use the notation r_i instead of $r_{u,i}$ in Equation 1); otherwise, we say that the platform is *heterogeneous*.

Finally, we let $r_{comm,i} = e^{-\lambda_\ell \, o_i \, / \, b}$ denote the reliability of the i-th communication.

E. Replication model

We use *spatial redundancy* to increase the reliability of a system: in other words, we replicate the intervals on several processors. Figure 3 shows an example of mapping by interval with spatial redundancy: the interval I_1 is mapped on the processors $\{P_1, P_2, P_3\}$, the interval I_2 is mapped on the processors $\{P_4, P_5\}$, and so on until the interval I_m, which is mapped on the processors $\{P_{p-1}, P_p\}$. Concerning the communications, the data-dependency o_{l_1} is mapped on the point-to-point links $\{L_{14}, L_{15}, L_{24}, L_{25}, L_{34}, L_{35}\}$, and so on.

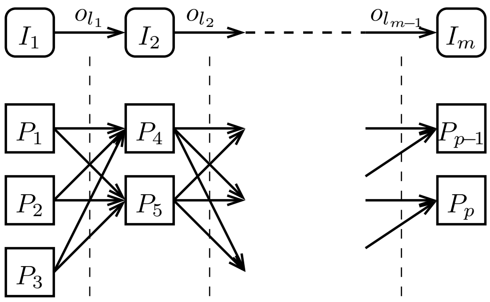

Figure 3. An example of interval mapping.

To increase the reliability, *each* processor of a given interval communicates with *each* processor of the next interval. Specifically, for any $1 \leq j \leq m - 1$, all the processors executing interval I_j send their result to all the processors executing the next interval I_{j+1}. Because of the bounded number $\mathcal{K}$ of possible communications (see Section II-B), the maximum number of replicas per interval is also limited to $\mathcal{K}$.

F. Multiprocessor mapping problem

We study several variants of the multiprocessor interval mapping problem. The inputs of the problem are a chain of n tasks $\mathcal{C} = (\tau_i)_{1 \leq i \leq n}$, a hardware platform of p processors $\mathcal{P} = (P_u)_{1 \leq u \leq p}$, and a bound $\mathcal{K}$ on the maximal number of replications for each interval of tasks. The output is an interval mapping of $\mathcal{C}$ onto $\mathcal{P}$, that is, a distribution of $\mathcal{C}$ into m intervals and an assignment of each interval to at most $\mathcal{K}$ processors of $\mathcal{P}$, such that each processor executes only one interval. Each variant of the mapping problem optimizes a different set of criteria among the following:

- the reliability,
- the expected input-output latency,
- the worst-case input-output latency,
- the expected period,
- the worst-case period.

Our contribution is multifold. In Section IV, we show how to compute the different objectives (reliability, expected and worst-case latency, and expected and worst-case period) for a given multiprocessor mapping. Then, for homogeneous platforms, we prove that:

1) computing a mono-criterion mapping that optimizes the reliability is *polynomial* (Section V-A);
2) optimizing both the reliability and the period remains *polynomial* (Section V-B);
3) the problem of optimizing both the reliability and the latency is *NP-complete* (Section V-C).

For heterogeneous platforms, we prove that optimizing the reliability only is *NP-complete*, and hence all the multi-criteria mapping problems that include the reliability in their criteria are also *NP-complete* (Section VI). Finally, we state some concluding remarks in Section VII.

III. Related work

Several papers have dealt with workflow applications the dependence graph of which is a linear chain. The pioneering papers [23], [24] investigate bi-criteria (period, latency) optimization of such workflows on homogeneous platforms. An extension of these results to heterogeneous platforms is provided in [5], [6].

All the previous papers deal with fully reliable platforms. In our previous work [4], we studied the (reliability, latency) mapping problem with fail-silent processors. The model in [4] is quite different, and much more crude, than the one of this paper: each processor has an absolute probability of failing, independent of task durations, and the faults are unrecoverable. To the best of our knowledge, we are not aware of other published work on optimizing linear chain workflows for reliability. However, many papers have dealt with a directed acyclic graph (DAG) instead of a pipelined workflow, be it a fully general DAG [8], a linear chain [20], or even independent tasks [15], [20]. The closest to our present work is [20]: it contains a short section on linear chains, with a mono-criterion dynamic programming algorithm for optimizing the reliability, which is similar to our Algorithm 1 (see Section V-A).

Finally, the specific problem of bi-criteria (length, reliability) multiprocessor scheduling has also been addressed in [7], [1], [13], [19], [11], [12] for general DAGs of operations, but except [1], [11], [12], these papers do not replicate the operations and have thus a very limited impact on the reliability. Moreover, none of them consider chains of tasks and interval mappings, and therefore they attempt to minimize the length of the mapping without distinguishing between the period and the latency (the latter one being equivalent to the schedule length).

IV. Evaluation of a given mapping

In this section, we detail the computation of the different objectives (reliability, expected and worst-case latency, and expected and worst-case period) for a given mapping. We compute the reliability of a mapping by building its *reliability block diagram* (RBD) [18], [3]. Formally, a RBD is an *acyclic oriented graph* (N, E), where each node of N is a *block* representing an element of the system, and each arc of E is a *causality link* between two blocks. Two particular connection points are its *source* S and its *destination* D. An RBD is *operational* if and only if there exists at least one operational path from S to D. A path is operational if and only if all the blocks in this path are operational. The probability that a block be operational is its reliability. By construction, the probability that a RBD is operational is equal to the reliability of the system that it represents.

In our case, the system is the multiprocessor interval mapping, possibly partial, of the application on the platform. A mapping is *partial* if not all intervals have been mapped yet, but of course those intervals that are mapped are such that all their predecessors are also mapped. Each block of the RBD represents an interval I_j placed on a processor or a data-dependency o_{l_j} between the two intervals I_j and I_{j+1} placed on a communication link. The reliability of a block is therefore computed according to Equation (2).

Computing the reliability in this way assumes that the occurrences of the failures are statistically independent events (see Section II-D). Without this hypothesis, the fact that some blocks belong to several paths from S to D makes the computation of the reliability very complex. Concerning hardware faults, this hypothesis is reasonable, but this would not be the case for software faults [17].

The main drawback of this approach is that the computation of the reliability is, in general, exponential in the size of the RBD. When the schedule is without replication, the RBD is *serial* (i.e., there is a single path from S to D) so the computation of the reliability is linear in the size of the RBD. But when the schedule is with replications, the RBD has no particular form, so the computation of the reliability is exponential in the size of the RBD. The reason is that processors are heterogeneous: the completion dates of a given interval on its assigned processors are different, so the reception dates by the processors of the next interval are different. This is true even when the application is a chain of intervals rather than a general graph. See Figure 4 for an illustration, where the RBD corresponding to the mapping has no specific form.

One solution for computing the reliability of the mapping of Figure 4 involves enumerating all the *minimal cut sets* of its RBD [16]. A *cut set* in a RBD is a set of blocks C such that there is no path from S to D if all the blocks of C are removed from the RBD. A cut C is *minimal* if, whatever the block that is removed from it, the resulting set is not a cut anymore. It follows that the reliability of a minimal cut set is the reliability of all its blocks put in parallel. The reliability of the mapping can then be approximated by the reliability of the alternative RBD composed of all the minimal cut sets put in sequence. Because this RBD is *serial-parallel*, this computation is linear in the number of minimal cut sets. The problem is that, in general, the number of minimal cuts is exponential in the size of the RBD [16].

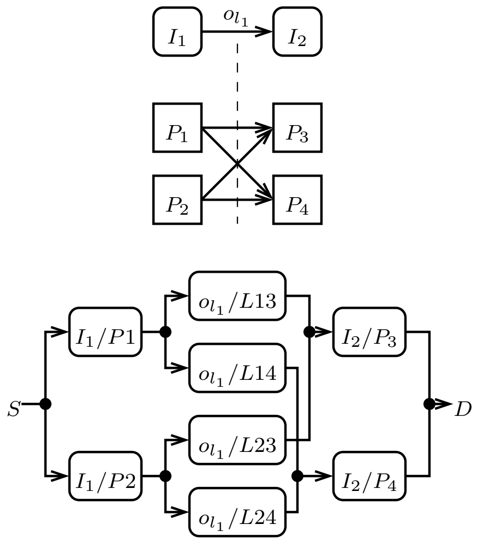

Figure 4. A mapping of two intervals (I_1 and I_2) on four processors (P_1 to P_4) and its RBD which has no particular form.

For this reason, we follow the approach of [11] and we insert *routing operations* between the intervals to make sure that the RBD representing a mapping is always *serial-parallel*, therefore making tractable the computation of the reliability. This is illustrated in Figure 5, where a routing operation R has been mapped on processor P_5 and the RBD corresponding to the mapping is serial-parallel; as a consequence, the reliability of this mapping can be computed in a linear time w.r.t. the number of intervals.

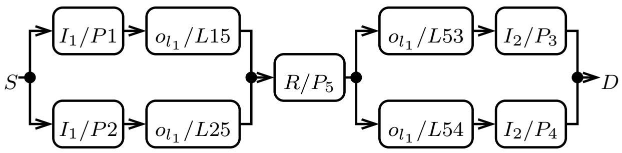

Figure 5. The serial-parallel RBD obtained from the same mapping as in Figure 4 but with an additional routing operation R.

Routing operations can be mapped on *any* processor. For instance, in the RBD of Figure 5, R could have been mapped on P_1 instead of P_5, therefore avoiding the need for the communication (o_{l_1}/L_{15}). Also, routing operations are assumed to be executed in 0 time units [11]. As a consequence, for any processor P_u, the reliability of the block (R/P_u) is 1.

As we have advocated, inserting routing operations yields the huge advantage of making the reliability computation linear in time. This comes at a cost in the execution time of the system because of the increased number of communications. For instance, in Figure 5, o_{l_1} is transmitted twice before reaching I_2. However, it has been shown in [11] that the overhead incurred by the routing operations is reasonable (only $+3.88\,\%$ on average).

For an interval I of weight W mapped on the subset of processors $\mathcal{P}_I$, let ec be its expected time of computation, and let wc be its WCET (by the slowest processor of $\mathcal{P}_I$). Assume that the processors in $\mathcal{P}_I$ are ordered according to their speed, from the fastest P_1 to the slowest P_t: that is, $\forall 1 \leq u < t$, we have $s_u \geq s_{u+1}$. Then, the expected and worst-case execution times of I on $\mathcal{P}_I$ are:

$$ec(I, \mathcal{P}_I) = W \times \frac{\sum_{u=1}^{t} \left(\frac{1}{s_u} \, r_{u,I} \prod_{v=1}^{u-1} (1 - r_{v,I}) \right)}{1 - \prod_{u=1}^{t} (1 - r_{u,I})} \tag{3}$$

$$wc(I, \mathcal{P}_I) = \frac{W}{s_t} \tag{4}$$

Equation (3) sums up, for each P_u, the case where the first $u-1$ fastest processors fail, and the u-th one is successful. Then, for a mapping $(I_1, \mathcal{P}_1), \ldots, (I_m, \mathcal{P}_m)$, the expected latency EL and the expected period EP are:

$$EL = \sum_{i=1}^{m} ec(I_i, \mathcal{P}_i) + \frac{o_i}{b} \quad (5)$$

$$EP = \max\{\max_{1 \le i \le m}\{\frac{o_i}{b}\}, \max_{1 \le i \le m}(ec(I_i, \mathcal{P}_i)\} \quad (6)$$

The worst-case latency WL and the worst-case period WP are defined similarly, but with the worst-case cost of intervals (Equation (4)) instead of the expected cost (Equation (3)):

$$WL = \sum_{i=1}^{m} wc(I_i, \mathcal{P}_i) + \frac{o_i}{b} \quad (7)$$

$$WP = \max\{\max_{1 \le i \le m}\{\frac{o_i}{b}\} \max_{1 \le i \le m}(wc(I_i, \mathcal{P}_i)\} \quad (8)$$

Finally, thanks to the routing operations, the reliability of the mapping $(I_1, \mathcal{P}_1), \ldots, (I_m, \mathcal{P}_m)$ is:

$$r = \prod_{i=1}^{t}\left(1 - \prod_{P_u \in \mathcal{P}_i}(1 - r_{comm,i-1} \times r_{u,I_i} \times r_{comm,i})\right) \quad (9)$$

Equation (9) above is computed according to the generic form of the RBD of Figure 5. To account for the fact that the first interval I_1 has no incoming communication, we just set $o_0 = 0$, hence $r_{comm,0} = 1$. The same occurs for the outgoing communication of the last interval I_m. Finally, routing operations do not appear in Equation (9) since their reliability is always equal to 1.

V. Complexity results for homogeneous platforms

In this section, we provide optimal polynomial algorithms for the mono-criterion reliability optimization problem, and then for the bi-criteria (reliability, period) optimization problem. Finally, we prove the NP-completeness of the bi-criteria (reliability, latency) optimization problem. Note that on homogeneous platforms, the expected latency and worst-case latency are the same. This also holds true for the expected period and worst-case period.

A. Reliability optimization

We present a mono-criterion polynomial-time algorithm that maximizes the reliability of a given chain of tasks on a given homogeneous platform. Algorithm 1 is a dynamic programming algorithm. It is a simplified version of Algorithm 2 for bi-criteria (reliability, period) optimization, which we present in the next section.

Data: a number p of fully homogeneous processors of failure rate λ, a list A of n tasks of sizes w_i, and a maximal number $\mathcal{K}$ of replications
Result: a reliability r

for $k = 1$ **to** $\min\{\mathcal{K}, p\}$ **do**
  $F(1, k) = 1 - (1 - r_{comm,0} \times r_1 \times r_{comm,1})^k$;
end
$F(0, 0) = 1$;
for $i = 1$ **to** n **do**
  $F(i, 0) = 0$;
end
for $i = 2$ **to** n **do**
  for $k = i$ **to** p **do**
    $F(i, k) = \max_{1 \le j < i, 1 \le q \le \min\{\mathcal{K},k\}} \left\{F(j, k-q) \times \left(1 - \left(1 - r_{comm,j-1} \times \prod_{j \le l \le i} r_l \times r_{comm,i}\right)^q\right)\right\}$;
  end
end
$r = \max_{1 \le q \le p} F(n, q)$;

Algorithm 1: Optimal algorithm for reliability optimization on fully homogeneous platforms.

Theorem 1. *Algorithm 1 computes in time $O(n^2p^2)$ the optimal mapping for reliability optimization on fully homogeneous platforms.*

Proof: In this algorithm, $F(i, k)$ is the optimal reliability when mapping the first i tasks on k processors, and it is computed iteratively with the dynamic programming procedure. ■

B. Reliability/period optimization

We now present a bi-criteria (reliability, period) polynomial-time algorithm that optimizes the reliability of a mapping given a bound on the period. Recall that, for homogeneous platforms, the worst-case period and the expected period are the same.

Theorem 2. *Algorithm 2 computes in time $O(n^2p^2)$ the optimal mapping for reliability optimization on fully homogeneous platforms, when a bound on the period is given.*

Proof: In this algorithm, $F(i, k)$ is again the optimal reliability when mapping the first i tasks on k processors. The dynamic programming procedure of Algorithm 1 has been modified to account for the period bound. ■

Finally, we observe that the converse problem, namely optimizing the period when a bound on the reliability is enforced, is polynomial too. It suffices to use a binary search on the period and to repeatedly execute Algorithm 2 until the optimal value is found.

Data: a number p of fully homogeneous processors of failure rate λ, a list A of n tasks of sizes w_i, a maximal number $\mathcal{K}$ of replications, and an upper-bound P on the period

Result: a reliability r

for $k = 1$ **to** $\min\{\mathcal{K}, p\}$ **do**
  if $\max\{o_0/b, w_1/s, o_1/b\} \leq P$ **then**
    $F(1,k) = \left(1-(1-r_{comm,0}\times r_1 \times r_{comm,1})^k\right)$;
  else
    $F(1,k) = 0$;
  end
end
for $i = 1$ **to** n **do**
  $F(i, 0) = 0$;
end
for *i=2* **to** n **do**
  for *k=i* **to** p **do**
    $F(i,k) = \max_{1\leq j<i, 1\leq q\leq \min\{\mathcal{K},k\}} \left\{F(j,k-q)\times \left(1-\left(1-r_{comm,j}\times \prod_{j<l\leq i} r_l \times r_{comm,i}\right)^q\right) \middle| \max\{o_j, \sum_{v=j+1}^{i} w_v, o_i\} \leq P\right\}$;
  end
end
$r = \max_{1\leq q\leq p} F(n,q)$;

Algorithm 2: Optimal algorithm for reliability optimization on fully homogeneous platforms, when a bound on the period is given.

C. Reliability/latency optimization

We now prove the NP-completeness of the bi-criteria (reliability, latency) optimization problem on homogeneous platforms. As for the period, there is no difference between the worst-case latency and the expected latency on such platforms.

Theorem 3. *The problem of optimizing the reliability on homogeneous platforms, with a bound on the latency, is NP-complete.*

Proof: Consider the associated decision problem: given an homogeneous platform, a chain of tasks, a bound $\mathcal{K}$ on the number of replications, a reliability r, and a latency L, does there exist a mapping whose reliability is at least r and whose latency is at most L? This problem is obviously in NP: given a mapping, it is easy to compute its reliability and latency, and to check that it is valid in polynomial time.

To establish the completeness, we use a reduction from 2-PARTITION (instance $\mathcal{I}_1$): given a set A of n numbers $a_1, \ldots, a_n$, does there exist a subset $A' \subset A$ such that $\sum_{a\in A'} a = \sum_{a\notin A'} a$. Let $T = \frac{1}{2}\sum_{a\in A} a$. Let $a_{min} = \min_{1\leq i\leq n}\{a_i\}$ and $a_{max} = \max_{1\leq i\leq n}\{a_i\}$. We build the following instance $\mathcal{I}_2$ of our problem with $3n+1$ tasks and $6n$ identical processors:

- $\mathcal{K} = 2$ and $\lambda = 10^{-8}10^{-n}a_{max}^{-3n}$;
- $s = b = 1$ (unit processor speed and link bandwidth);
- $B = \frac{1}{2a_{min}}\left(\frac{n}{4} + na_{max}^2 + T + 2\right)$;
- $\forall 1 \leq i \leq n$, $w_{3i-2} = B$, $w_{3i-1} = \frac{1}{2}$ and $w_{3i} = a_i$;
- $w_{3n+1} = B$;
- $\forall 1 \leq i \leq n, r_i = e^{-\lambda w_i}$ and $r_{comm,i} = 1$;
- $\forall 1 \leq i \leq n$, $o_{3i-2} = 0$, $o_{3i-1} = a_i$ and $o_{3i} = 0$;
- $L = (n+1)B + \frac{n}{2} + 3T$;
- it follows that the reliability of the mapping is $r = (1-(1-e^{-\lambda B})^2)^{n+1}\times(1-\lambda^2(\frac{n}{4} + \sum_{1\leq i\leq n} a_i^2 + T) - \lambda^4 \times 2^{2n}(a_{max}+1)^n)$.

The size of instance $\mathcal{I}_2$ is polynomial in the size of $\mathcal{I}_1$. We now show that $\mathcal{I}_1$ has a solution if and only if $\mathcal{I}_2$ has a solution. Suppose first that $\mathcal{I}_1$ has a solution A'. Then we propose the following solution for $\mathcal{I}_2$:

- all intervals are replicated 2 times;
- any task of size B make up an interval;
- for all $1 \leq i \leq n$, if $a_i \in A'$, then T_{3i-1} and T_{3i} are assigned to two different intervals, else they constitute one single interval.

This yields the following costs for the latency:

- the sum of computation costs does not depend of the mapping: $(n+1)B + \frac{n}{2} + 2T$;
- for each $a_i \in A'$, we add a communication cost a_i.

We thus obtain a latency $L = (n+1)B + \frac{n}{2} + 3T$. Concerning the reliability, it is the product of the reliability of all intervals:

- the reliability of intervals of size B is $(1-(1-e^{-\lambda B})^2)$;
- for each $a_i \in A'$, the product of the reliability of the two intervals for tasks T_{3i-1} and T_{3i-1} is $(1-(1-e^{-\frac{\lambda}{2}})^2)(1-(1-e^{-\lambda a_i})^2)$, which is greater than $(1-\frac{\lambda^2}{4})(1-\lambda^2 a_i^2)$;
- for each $a_i \notin A'$, the reliability of the interval for tasks T_{3i-1} and T_{3i-1} is $(1-(1-e^{-\lambda(a_i+\frac{1}{2})})^2)$, which is greater than $1-\lambda^2(a_i+\frac{1}{2})^2$.

We thus obtain, for the product of all these reliabilities,

$$\begin{aligned}
r' &= (1-(1-e^{-\lambda B})^2)^n \times \\
&\quad \prod_{a_i\in A'}(1-(1-e^{-\frac{\lambda}{2}})^2)(1-(1-e^{-\lambda a_i})^2) \times \\
&\quad \prod_{a-i\notin A'}\left(1-\left(1-e^{-\lambda\left(a_i+\frac{1}{2}\right)}\right)^2\right) \\
&\geq (1-(1-e^{-\lambda B})^2)^n \times \\
&\quad \prod_{a_i\in A'}(1-\tfrac{\lambda^2}{4})(1-\lambda^2 a_i^2) \times \\
&\quad \prod_{a-i\notin A'}(1-\lambda^2(a_i+\tfrac{1}{2})^2) \\
&\geq (1-(1-e^{-\lambda B})^2)^n \times \\
&\quad \left(1-\lambda^2\left(\tfrac{n}{4}+\textstyle\sum_{1\leq i\leq n}a_i^2+T\right)-\lambda^4 2^{2n}(a_{max}+1)^n\right)
\end{aligned}$$

Suppose now that $\mathcal{I}_2$ has a solution. The exponent in the reliability bound implies that any interval is replicated at least 2 times, and the bound on replication is 2. This means that all intervals are replicated exactly 2 times. Suppose that one of the tasks of size B is computed together with another task in the same interval. This yields the bound on reliability:

$$\begin{aligned}
r' &< (1-(1-e^{-\lambda B})^2)^n(1-(1-e^{-\lambda(B+a_{min})})^2) \\
&< (1-(1-e^{-\lambda B})^2)^{n+1}\times\tfrac{1-\lambda^2(B+a_{min})^2}{1-\lambda^2B^2(1-\frac{\lambda B}{2})^2} \\
&< (1-(1-e^{-\lambda B})^2)^{n+1}(1-\lambda^2(B+a_{min})^2) \\
&\quad (1+\lambda^2B^2(1-\tfrac{\lambda B}{2})^2+2\lambda^4B^4(1-\tfrac{\lambda B}{2})^4) \\
&< (1-(1-e^{-\lambda B})^2)^{n+1}\times(1-2\lambda^2Ba_{min}+7\lambda^4B^4) \\
&< r
\end{aligned}$$

This means that any task of size B makes up an interval. Let A' be the set of values i such that T_{3i-1} and T_{3i} are not in the same interval. We obtain the following formulas:

- For the reliability:

$$\begin{aligned}
r &\leq (1-(1-e^{-\lambda B})^2)^n \times \\
&\quad \prod_{a_i\in A'}(1-(1-e^{-\frac{\lambda}{2}})^2)(1-(1-e^{-\lambda a_i})^2) \times \\
&\quad \prod_{a_i\notin A'}\left(1-\left(1-e^{-\lambda\left(a_i+\frac{1}{2}\right)}\right)^2\right) \\
&\leq (1-(1-e^{-\lambda B})^2)^n \times \\
&\quad \prod_{a_i\in A'}(1-\tfrac{\lambda^2}{4}(1-\tfrac{\lambda}{4})^2)(1-\lambda^2a_i^2(1-\lambda a_i)^2 \times \\
&\quad \prod_{a_i\notin A'}(1-(\lambda^2+\tfrac{\lambda^2}{4}+\lambda^2a_i)(1-\tfrac{\lambda}{2}(a_i+\tfrac{1}{2}))^2) \\
&\leq 1-\lambda^2(\tfrac{n}{4}+\textstyle\sum_{1\leq i\leq n}a_i^2+\sum_{a_i\notin A'}a_i)+\lambda^3 10^n a_{max}^{3n}
\end{aligned}$$

- For the latency:

$$(n+1)B+\frac{n}{2}+\sum_{a_i\in A'}a_i+2T\leq(n+1)B+\frac{n}{2}+3T$$

This means $\sum_{a_i\notin A'}a_i\leq T$ and $\sum_{a_i\in A'}a_i\leq T$. Hence, A' is a solution for $\mathcal{I}_1$. This concludes the proof. ∎

We conclude that, on homogeneous platforms, the bi-criteria (reliability, period) problem is polynomial, while the bi-criteria (reliability, latency) problem is NP-complete. As a consequence, the tri-criteria (reliability, period, latency) problem is NP-complete too.

It is striking, and somewhat unexpected, that the bi-criteria (reliability, period) problem is easier than the (reliability, latency) one. The intuition for this difference is the following: when the period bound is given, we know once and for all which processors are fast enough to be enrolled for a given interval. Therefore, the mapping choices are local. On the contrary, the computation of the latency remains global, and its final value, including communication costs, depends upon the choices that will be made further on.

VI. Complexity results for heterogeneous platforms

In this section, we prove the NP-completeness of the reliability optimization problem on heterogeneous platforms.

Theorem 4. *The problem of optimizing the reliability on heterogeneous platforms is NP-complete.*

Proof: Consider the associated decision problem: given a heterogeneous platform, a chain of tasks, a bound on the number $\mathcal{K}$ of replications, and a reliability r, does there exist a mapping of reliability at least r? This problem is obviously in NP: given a reliability and a mapping, it is easy to compute the reliability and to check that it is valid in polynomial time. To establish the completeness, we use a reduction from 3-PARTITION. Consider the following general instance $\mathcal{I}_1$ of 3-PARTITION: given $3n$ numbers $a_1,\ldots,a_{3n}$ and a number T such that $\sum_{1\leq j\leq 3n}a_j=nT$, does there exist n independent subsets $B_1,\ldots,B_n$ of $\{a_1,\ldots,a_{3n}\}$ such that for all $1\leq i\leq n$, $\sum_{a_j\in B_i}a_j=T$? Let $a_{min}=\min_{1\leq i\leq 3n}i\{a_i\}$.

We build the following instance $\mathcal{I}_2$ with n tasks and $p=3n$ processors:

- $\lambda=\frac{10^{-8}}{nT^2}$;
- $\mathcal{K}=3$;
- $\gamma=1+\frac{1}{2(T-1)}$;
- $\forall 1\leq i\leq n, w_i=1/n$ (all tasks have cost $1/n$);
- $r_{u,i}=e^{-\lambda_u\frac{w_i}{s_u}}$;
- $r_{comm,i}=1$;
- $\forall 1\leq u\leq 3n, \lambda_u=\lambda*\gamma^{a_u}$ and $s_u=1$;
- it follows that the reliability of the mapping is $r=\left(1-\lambda^3\gamma^T\right)^n$.

The size of $\mathcal{I}_2$ is polynomial in the size of $\mathcal{I}_1$. We show that $\mathcal{I}_1$ has a solution if and only if $\mathcal{I}_2$ has a solution.

Suppose first that $\mathcal{I}_1$ has a solution $B_1,\ldots,B_n$. We propose the following solution for $\mathcal{I}_2$:

- we have one interval per task;
- the i-th task is replicated three times and allocated to the set of processors $\{P_u | u \in B_i\}$.

We obtain the following reliability for task i:

$$(1 - \prod(1 - e^{-\lambda\gamma^{a_i}})) \geq 1 - \prod(\lambda\gamma^{a_i}) \geq 1 - \lambda^3\gamma^T ,$$

Hence, the overall reliability is $r \geq (1 - \lambda^3\gamma^T)^n$.

Suppose now that $\mathcal{I}_2$ has a solution. We first show that the optimal mapping consists of n intervals, one per task, each replicated three times. Suppose that we know the number of intervals in the optimal mapping. There are at most n intervals, and we have enough processors to replicate all of them three times, and this increases the reliability. We conclude that all intervals will be replicated three times. Suppose now that one of these intervals contains $t > 1$ tasks. There are enough processors to split this interval into t single-task intervals, each replicated three times. Let r_1 be the reliability of the original interval with t tasks, and r_t the reliability of the same tasks assigned to t intervals replicated three times. By hypothesis of optimality, we have:

$$\begin{array}{llll}
 & r_1 & \geq & r_t \\
\Rightarrow & e^{-\lambda\gamma t} & \geq & 1 - (1 - e^{-\lambda\gamma^T})^t \\
\Rightarrow & \lambda\gamma t - \frac{1}{2}(\lambda\gamma t)^2 & \leq & (\lambda\gamma^T)^t \qquad \text{because } \lambda\gamma^T \leq 1 \\
\Rightarrow & \lambda\gamma 2 - \frac{1}{2}(\lambda\gamma 2)^2 & \leq & (\lambda\gamma 2)^2 \qquad \text{because } \gamma^{T-1} \leq 2 \\
\Rightarrow & \lambda\gamma 2 & \leq & \frac{3}{2}(\lambda\gamma 2)^2 \\
\Rightarrow & \lambda\gamma 2 & \geq & \frac{2}{3} \\
\Rightarrow & 4\lambda & \geq & \frac{2}{3}
\end{array}$$

However, $\lambda \leq 10^{-8}$, which contradicts the hypothesis. This means that, in the optimal solution, any task constitutes an interval.

Let, for all i, $B_i = \{a_j, T_i \text{ mapped on } P_j\}$. We obtain the following reliability:

$$r = \prod_{1 \leq i \leq n} (1 - \prod_{a_j \in B_i} (1 - e^{-\lambda\gamma^{a_i}})) \geq (1 - \lambda^3\gamma^T)^n.$$

Suppose that, for a value i, $\sum_{a_j \in B_i} a_j \neq T$. Then,

$$\begin{aligned}
r &\leq \textstyle\prod_{1 \leq i \leq n} (1 - \prod_{a_j \in B_i} (\lambda\gamma^{a_i} - \frac{1}{2}(\lambda\gamma^{a_i})^2)) \\
&\leq \textstyle\prod_{1 \leq i \leq n} (1 - \lambda^3\gamma^{\sum_{a_j \in B_i} a_j} \prod_{a_j \in B_i} (1 - \frac{1}{2}\lambda\gamma^{a_i})) \\
&\leq \textstyle\prod_{1 \leq i \leq n} (1 - \lambda^3\gamma^{\sum_{a_j \in B_i} a_j} (1 - \frac{\lambda}{2} \sum_{a_j \in B_i} \gamma^{a_j})) \\
&\leq \textstyle\prod_{1 \leq i \leq n} (1 - \lambda^3\gamma^{\sum_{a_j \in B_i} a_j} (1 - \frac{3\lambda}{2}\gamma^T)) \\
&\leq \textstyle\prod_{1 \leq i \leq n} (1 - \lambda^3\gamma^{\sum_{a_j \in B_i} a_j} + \frac{3\lambda^4}{2}\gamma^{T + \sum_{a_j \in B_i} a_j}) \\
&\leq \textstyle\prod_{1 \leq i \leq n} (1 - \lambda^3\gamma^{\sum_{a_j \in B_i} a_j})(1 + \frac{\frac{3\lambda^4}{2}\gamma^{T + \sum_{a_j \in B_i} a_j}}{1 - \lambda^3\gamma^{\sum_{a_j \in B_i} a_j}}) \\
&\leq \textstyle\prod_{1 \leq i \leq n} (1 - \lambda^3\gamma^{\sum_{a_j \in B_i} a_j})(1 + \frac{\frac{3\lambda^4}{2}\gamma^{4T}}{1 - \lambda^3\gamma^{3T}}) \\
&\leq (1 + \frac{\frac{3\lambda^4}{2}\gamma^{4T}}{1 - \lambda^3\gamma^{3T}})^n \times \textstyle\prod_{1 \leq i \leq n} (1 - \lambda^3\gamma^{\sum_{a_j \in B_i} a_j})
\end{aligned}$$

By hypothesis, we have $\sum_{a_j \in B_i} a_j \neq T$ for a value i. Then by convexity,

$$\prod_{1 \leq i \leq n} (1 - \lambda^3\gamma^{\sum_{a_j \in B_i} a_j}) \leq$$
$$(1 - \lambda^3\gamma^T)^{n-2} \times (1 - \lambda^3\gamma^{T-1}) \times (1 - \lambda^3\gamma^{T+1})$$

By hypothesis, we have:

$$\begin{array}{rcl}
(1 - \lambda^3\gamma^T)^n & \leq & r \\
 & \leq & (1 + \frac{\frac{3\lambda^4}{2}\gamma^{4T}}{1 - \lambda^3\gamma^{3T}})^n (1 - \lambda^3\gamma^T)^{n-2} \\
 & & (1 - \lambda^3\gamma^{T-1})(1 - \lambda^3\gamma^{T+1}) \\
\Rightarrow (1 - \lambda^3\gamma^T)^2 & \leq & (1 + \frac{\frac{3\lambda^4}{2}\gamma^{4T}}{1 - \lambda^3\gamma^{3T}})^n \\
 & & (1 - \lambda^3\gamma^{T-1})(1 - \lambda^3\gamma^{T+1}) \\
 & \leq & (1 + \frac{\frac{3\lambda^4}{2}\gamma^{4T}}{1 - \lambda^3\gamma^{3T}})^n \\
 & & \left((1 - \lambda^3\gamma^T)^2 - \lambda^3\gamma^{T-1}(\gamma - 1)^2\right) \\
\Rightarrow (1 - \lambda^3\gamma^T)^2 & \geq & \left((1 + \frac{\frac{3\lambda^4}{2}\gamma^{4T}}{1 - \lambda^3\gamma^{3T}})^n - 1\right)^{-1} \\
 & & (1 + \frac{\frac{3\lambda^4}{2}\gamma^{4T}}{1 - \lambda^3\gamma^{3T}})^n \lambda^3\gamma^{T-1}(\gamma - 1)^2 \\
 & \geq & \frac{1 + \frac{3\lambda^4}{4}n\gamma^{4T}}{\frac{3\lambda^4}{4}n\gamma^{4T}} \lambda^3\gamma^{T-1}(\gamma - 1)^2 \\
 & \geq & \frac{1 + \frac{3\lambda^4}{4}n\gamma^{4T}}{3\lambda n\gamma^{3T+1}(T-1)^2}
\end{array}$$

However, $3\lambda n\gamma^{3T+1}(T - 1)^2 \leq 1$ and $1 + \frac{3\lambda^4}{4}n\gamma^{4T} \geq 1$. This contradicts the hypothesis. Then, if $\{B_1, \ldots, B_n\}$ corresponds to a solution of $\mathcal{I}_2$, we have $\sum_{a_j \in B_i} a_j = T$ for $1 \leq i \leq n$. This shows that $B_1, \ldots, B_n$ is a solution for $\mathcal{I}_1$, which concludes the proof. ■

Because mono-criterion reliability optimization is already NP-complete, all multi-criteria problems with period or latency or both, are also NP-complete on heterogeneous platforms.

VII. Conclusion

We have addressed problems related to the mapping of linear chain workflows on homogeneous and heterogeneous distributed platforms. The main goal was to optimize the reliability of the mapping through task replication, while enforcing bounds on performance-oriented criteria (period and latency). We have been able to derive a comprehensive set of NP-hardness complexity results, together with optimal algorithms for polynomial instances. Altogether, these results provide a solid theoretical foundation for the study of multi-criteria mappings of linear chain workflows. Another contribution of this paper is the introduction of a realistic communication model that nicely accounts for the inherent physical limitations on the communication capabilities of state-of-the-art processors.

Communication failures have been incorporated in the model through routing operations, which guarantee that evaluating the system reliability remains computationally tractable. An interesting future research direction would be to investigate whether it is feasible to remove this routing procedure, and accurately approximate the reliability of general systems (non serial-parallel).

Another direction for future work involves the design of efficient heuristics for even more difficult problems that would mix performance-related criteria (period, latency) with several other objectives, such as reliability, resource costs, and power consumption.

Acknowledgements

Anne Benoit and Yves Robert are with the Institut Universitaire de France. This work was supported in part by the ANR StochaGrid project, and by a Marie-Curie International Outgoing Fellowship within the 7th European Community Framework Programme.

References

[1] I. Assayad, A. Girault, and H. Kalla. A bi-criteria scheduling heuristics for distributed embedded systems under reliability and real-time constraints. In *Int. Conf. on Dependable Systems and Networks, DSN'04*, pages 347–356, Firenze, Italy, June 2004. IEEE.

[2] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Secure Comput.*, 1(1):11–33, Jan. 2004.

[3] H. Balaban. Some effects of redundancy on system reliability. In *National Symposium on Reliability and Quality Control*, pages 385–402, Washington (DC), USA, Jan. 1960.

[4] A. Benoit, V. Rehn-Sonigo, and Y. Robert. Optimizing latency and reliability of pipeline workflow applications. In *HCW'08, the 17th Heterogeneity in Computing Workshop*. IEEE Computer Society Press, 2008.

[5] A. Benoit and Y. Robert. Mapping pipeline skeletons onto heterogeneous platforms. *J. Parallel Distributed Computing*, 68(6):790–808, 2008.

[6] A. Benoit and Y. Robert. Complexity results for throughput and latency optimization of replicated and data-parallel workflows. *Algorithmica*, 57(4):689–724, 2010.

[7] A. Dogan and F. Özgüner. Matching and scheduling algorithms for minimizing execution time and failure probability of applications in heterogeneous computing. *IEEE Trans. Parallel and Distributed Systems*, 13(3):308–323, Mar. 2002.

[8] J. Dongarra, E. Jeannot, E. Saule, and Z. Shi. Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems. In *ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 280–288. ACM Press, 2007.

[9] C. Ferdinand, R. Heckmann, M. Langenbach, F. Martin, M. Schmidt, H. Theiling, S. Thesing, and R. Wilhelm. Reliable and precise WCET determination for a real-life processor. In *Int. Workshop on Embedded Software, EMSOFT'01*, volume 2211 of *LNCS*. Springer-Verlag, 2001.

[10] F. Gärtner. Fundamentals of fault-tolerant distributed computing in asynchronous environments. *ACM Computing Surveys*, 31(1):1–26, Mar. 1999.

[11] A. Girault and H. Kalla. A novel bicriteria scheduling heuristics providing a guaranteed global system failure rate. *IEEE Trans. Dependable Secure Comput.*, 6(4):241–254, Dec. 2009.

[12] A. Girault, E. Saule, and D. Trystram. Reliability versus performance for critical applications. *J. of Parallel and Distributed Computing*, 69(3):326–336, Mar. 2009.

[13] M. Hakem and F. Butelle. A bi-objective algorithm for scheduling parallel applications on heterogeneous systems subject to failures. In *Rencontres Francophones du Parallélisme, RENPAR'06*, Perpignan, France, Oct. 2006.

[14] B. Hong and V. K. Prasanna. Adaptive allocation of independent tasks to maximize throughput. *IEEE Trans. Parallel Distributed Systems*, 18(10):1420–1435, 2007.

[15] E. Jeannot, E. Saule, and D. Trystram. Bi-objective approximation scheme for makespan and reliability optimization on uniform parallel machines. In *Euro-Par*, volume 5168 of *Lecture Notes in Computer Science*, pages 877–886. Springer, 2008.

[16] P. Jensen and M. Bellmore. An algorithm to determine the reliability of a complex system. *IEEE Trans. Reliability*, 18:169–174, Nov. 1969.

[17] J. Knight and N. Leveson. An experimental evaluation of the assumption of independence in multi-version programming. *IEEE Trans. Software Engin.*, 12(1):96–109, 1986.

[18] D. Lloyd and M. Lipow. *Reliability: Management, Methods, and Mathematics*, chapter 9. Prentice-Hall, 1962.

[19] P. Pop, K. Poulsen, and V. Izosimov. Scheduling and voltage scaling for energy/reliability trade-offs in fault-tolerant time-triggered embedded systems. In *International Conference on Hardware-Software Codesign and System Synthesis, CODES+ISSS'07*, Salzburg, Austria, Oct. 2007. ACM.

[20] E. Saule and D. Trystram. Analyzing scheduling with transient failures. *Information Processing Letters*, 109(11):539–542, 2009.

[21] S. Shatz and J.-P. Wang. Models and algorithms for reliability-oriented task-allocation in redundant distributed-computer systems. *IEEE Trans. Reliability*, 38(1):16–26, Apr. 1989.

[22] J. Souyris, E. Pavec, G. Himbert, V. Jégu, G. Borios, and R. Heckmann. Computing the worst case execution time of an avionics program by abstract interpretation. In *International Workshop on Worst-case Execution Time, WCET'05*, pages 21–24, Mallorca, Spain, July 2005.

[23] J. Subhlok and G. Vondran. Optimal mapping of sequences of data parallel tasks. In *PPoPP'95*, pages 134–143. ACM Press, 1995.

[24] J. Subhlok and G. Vondran. Optimal latency-throughput tradeoffs for data parallel pipelines. In *SPAA'96*, pages 62–71. ACM Press, 1996.

[25] D. Zhu, R. Melhem, and D. Mossé. The effects of energy management on reliability in real-time embedded systems. In *International Conference on Computer Aided Design, ICCAD'04*, pages 35–40, San Jose (CA), USA, Nov. 2004.