
Performance of a $\log(n)$ Distributed Mutual Exclusion Algorithm in case of Non-Equiprobability of Processes Requests

Michel Tréhel* – Pierre Gradit** – Alain Giorgetti*

* Laboratoire d'Informatique de l'Université de Franche-Comté
16, route de Gray
25030 Besançon Cedex, France
{trehel,giorgetti}@lifc.univ-fcomte.fr

** Laboratoire d'Analyse et d'Architecture des Systèmes
7, avenue du Colonel Roche
31077 Toulouse Cedex, France
gradit@laas.fr

RÉSUMÉ. L'algorithme distribué d'exclusion mutuelle de Naimi-Tréhel est fondé sur une structure dynamique d'arbre enraciné : quand un processus demande la section critique, il envoie une requête qui chemine jusqu'à la racine de l'arbre, puis il devient la nouvelle racine de l'arbre réorganisé. Quand les demandes des processus sont équiprobables, l'algorithme ne requiert que H_{n-1} (de l'ordre de $\log n$) messages en moyenne, n étant le nombre de processus en réseau. Cet article étudie le cas de non-équiprobabilité et montre que le nombre de messages est toujours inférieur à H_{n-1} . De plus, il propose une relation entre la variance de la distribution des probabilités de requête et la complexité de l'algorithme.

ABSTRACT. The Naimi-Tréhel distributed mutual exclusion algorithm is based on a dynamic rooted tree: when a process requests the critical section, it sends a request that eventually reaches the root and becomes the new root in the reorganized tree. When the processes requests are equiprobabilistic, the algorithm requires only H_{n-1} messages (order of $\log n$) on average, where n is the number of processes in the network. This paper studies the case when the equiprobabilistic hypothesis does not run and shows that the number of messages is always less than H_{n-1} . It presents a relation between the variability of the set of probabilities of processes requests and the complexity.

MOTS-CLÉS : Algorithme distribué, exclusion mutuelle, performances, non-équiprobabilité, variables distribuées.

KEYWORDS : Distributed algorithm, mutual exclusion, performances, non-equiprobability, distributed variables.

1. Introduction

A resource, shared by a set of processes, is said to be in mutual exclusion access when only one process can use it at a time. A mutual exclusion algorithm warrants this access restriction. In a distributed system, the processes may communicate only by messages. A distributed mutual exclusion algorithm defines a message exchange policy in order to fulfil access requirements. The performance of a distributed mutual exclusion algorithm is evaluated by counting the number of messages generated by its policy. It is in general easier to count this number when processes request with the same probability. Nevertheless, the equiprobabilistic case is not the more realistic. For instance, on a network of computers, only the active computers will request the critical section. It is the reason why it is interesting to examine how the performances of an algorithm evolves with the requests probabilities. This paper is concerned by the Naimi-Tréhel mutual exclusion algorithm[7]. We do not modify the algorithm, in the version given in [3]. We evaluate the average performances if the probabilities of processes requests are distinct. Our result is to show that the Naimi-Tréhel algorithm is *as it is* load-balanced. The first conjecture we found to depict that interesting feature is :“equiprobability has the worst average-case”. In a second time, this conjecture is refined to a more precise one, which introduces a term of first order (i.e. the variance) in the bounding which highlights the load-balancing feature.

Let us present some distributed mutual exclusion algorithms to see the influence of the non-equiprobability of processes requests. In token ring algorithms [11], the token circulates on the ring of computers. When a process requests the critical section, it waits for the arrival of the token. If a process requests frequently the critical section, that does not modify the circulation of the token. In Lamport’s algorithm [4], a requesting process broadcasts a request message with a time-stamp. The return of time-stamped acknowledgements allows it to check whether a process has invoked the critical section earlier than itself. The number of exchanged messages is $3(n - 1)$, where n is the number of processes in the network. As for the token ring, the non-equiprobability does not modify the number of messages. Ricart and Agrawala [12] improve the Lamport algorithm. The number of messages is $2(n - 1)$. Carvalho and Roucairol [1] proposed a modification of the Ricart-Agrawala algorithm in order to reduce the number of messages when the same process requests the critical section several times. It is an interesting idea, not so far from our concern. Nevertheless, our study avoids a modification of the algorithm. Maekawa [6] introduced the notion of arbitrating processes. A process wishing resource access must obtain the permission from a fixed set of \sqrt{n} arbitrating processes. Each arbitrating node gives the permission on behalf of itself and $(\sqrt{n} - 1)$ other nodes. The message complexity of this algorithm is $3(\sqrt{n} - 1)$. Once more, this algorithm does not depend on the probability of nodes requests.

In the Naimi-Tréhel algorithm[7], the nodes are organized in a dynamic rooted tree, the root being the token allowing to enter the critical section. Each node i maintains a pointer “Father” that indicates the process to which requests should be forwarded. When a process x forward a request to its former father, the requesting process becomes the new father of the forwarding process (x). Each request hits even-

tually the root and is then allowed to enter the critical section. The message complexity of this algorithm is H_{n-1} in the average-case [5, 8]. An algorithm, merging this algorithm and Raymond's algorithm [10] preserving a specific tree structure (open cube) can warranty the worst-case to be $O(\log(n))$ [2]. The initial Naimi-Tréhel algorithm takes into consideration another pointer "next", useful when a node requests the critical section before the previous requester leaves the critical section. This pointer has no influence on the performances. We present the algorithm without the "next" variable, replacing it for liveness purposes by a blocking mechanism, as in [3].

This paper is organized as follows: In section 2 we introduce the algorithm, in section 3 we define the complexity, and in section 4 we study the consequences of the non-equiprobability.

2. Naimi-Tréhel algorithm presentation

This algorithm is based on the following principles:

- each non-requesting process has a father to whom it will send its critical section request.
- a non-requesting process forwards its incoming requests and shall set after each forwarding the forwarded request sender as new father.
- This "ascending request" eventually hits the process holding the token which sends it to the request sender and sets the request sender as new father.

Figure 2 depict this "step-by-step ascension" where requesting process is tagged with a box. Arrows represent both request (when coming from the tagged node) and father pointer (when coming from untagged nodes).

Media Hypothesis The system is composed of a set of processes which can send messages to any of them. A sent message is eventually received. We assume there is no order constraints between sending and receiving.

Algorithmic expression The language used is based on a "C"-like writing where each top-level instruction is atomic. For clarity, each top-level instruction is numbered. Two specific instructions are added [3]:

on receive $\langle\langle\text{message}\rangle\rangle$ heads a procedure automatically called on reception of $\langle\langle\text{message}\rangle\rangle$. This command structure was present in the original paper [7] and can be related to a *remote procedure call*.

wait $\langle\text{condition}\rangle$ **do** at the beginning of the line shows that the condition shall be fulfilled before the atomic execution of the body.

Process local variables The set of processes is called $Sites = [1..n]$. Each process has three local variables, which are the only variables used in the algorithm:

father : “reference” which refers to the process to whom a non requesting process will ask for the critical section ($father \in Sites$). A requesting process — i.e. who has sent a request — has no more father ($father = \text{nil}$). The process holding the *Token* (whether it uses or not) has the token as father ($father = \text{Token}$). Thus, the set of “References” is $Sites \cup \{\text{nil}, \text{Token}\}$.

requesting : **boolean** is set true while the “main” procedure is currently executed.

requests : **integer** indicating how many “on receive request” procedures are currently executed locally. The purpose of this counter is to preserve liveness and allows to drop the “next” pointer — as in [3] — which was used when a requesting process receives a request. Intuitively, this pointer is replaced by the policy : “Each process works in priority for the others”. This policy is implemented by the fact that any request blocks the local process request sending. Conversely, requesting blocks the “on receive request” procedure.

Initial state For the description of the initial state, we note $i.id$ the variable *id* of process *i*. At the initial state, each process is non-requesting and a process (e.g. 1) holds the token. The token holder is also the common father of all the other processes. More formally:

$$\begin{aligned} \forall i \in Sites, i.requesting &= false, \quad i.requests = 0. \\ 1.father &= Token, \quad \forall i \in Sites \setminus \{1\}, i.father = 1 \end{aligned}$$

Algorithm A process will invoke the “Main” procedure for each “critical” processing it plans. Figure 1 gives the whole algorithm composed of three routines.

3. Sequential complexity definition and equiprobabilistic case

This part first focuses on the meaning of complexity. This complexity could be the number of top-level instruction executions, but the reachable state graph would have many similar states without real external meaning.

The complexity used here for the Naimi-Tréhel algorithm is based on a “non-concurrent hypothesis”: it is the number of message exchanged to satisfy a request with no more than one request at a time [7]. We call this “sequential complexity” since the different requests can be considered in sequence. This complexity will allow us to define a simplified reachable state graph.

This section is organized as follows: the first subsection defines the notion of state, the second subsection defines the simplified reachable state graph according to the “non-concurrent hypothesis”, the third subsection details the complexity definition. The last subsections develop illustrative examples.

<pre> 1. Main(critical) { 1. wait (requests == 0) do { requesting = true; if (father ∈ Sites) then { send request(this) to father; father = nil; } endif; } 2. wait (father == Token) do critical; 3. requesting = false; } </pre>	<pre> 2. on receive Token { 1. father = Token } 3. on receive request(i) { 1. requests ++; 2. wait (requesting = false) do { if (father == Token) then send Token to i; else send request(i) to father; endif; father = i; } 3. requests --; } </pre>
---	---

Figure 1. Naimi-Tréhel Algorithm

3.1. The abstract tree structure

Following the approach of [3], let us define a tree structure whose set of nodes is $Sites \cup \{Token\}$ by drawing a box around requesting processes and drawing an arrow:

- $i \rightarrow j$ when
 - $i.father = j$,
 - or **request**(j) is processed by i .
- $i \rightarrow Token$ when
 - $i.father = Token$,
 - or **Token** message is processed by i .

The initial state is given by the graph $[2..n] \rightarrow 1 \rightarrow Token$. Thus the reachable state graph can be reduced according to events modifying the structure: 1.1 (box the process), 1.3 (remove the box) and 3.2 (request forwarding). Figure 2 details the ascension of a requesting process. Large arrows are tree transformations while thin arrow are tagging/untagging operations. Tree structure is preserved during the whole process “ascent” corresponding to the request forwarding until it hits the token. The number indicated on these arrows is the number of messages being sent during the ascent. Notice that the final number of messages (i.e. 3) is equal to the height of requesting node in the initial tree.

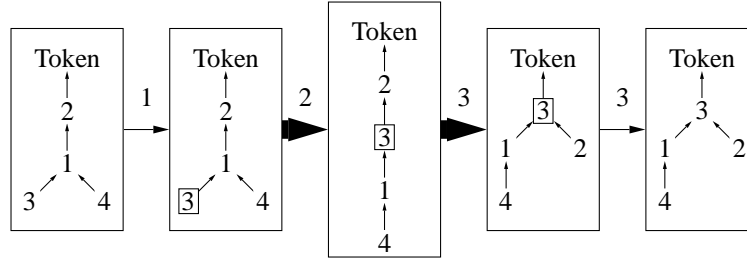


Figure 2. A process “ascent” step-by-step

3.2. Non-concurrent request hypothesis

Moreover, the non-concurrency hypothesis allows us to reduce the reachable state graph by considering only states with less than a single boxed process. The state sequence given in Figure 2 belongs to that reduced state graph. In fact these sequences have always the following form $(1.1).(3.2)^*.(1.3)$ where a **bold** transition represents transition where a message is emitted. Thus such a complete sequence can be abbreviated to a single labelled arrow relating two abstract trees without requesting processes. Figure 3 shows abbreviation of Figure 2. It results from previous considerations that the reachable state graph can be drastically simplified with the non-concurrency hypothesis in a graph whose nodes are abstract trees without requesting processes.

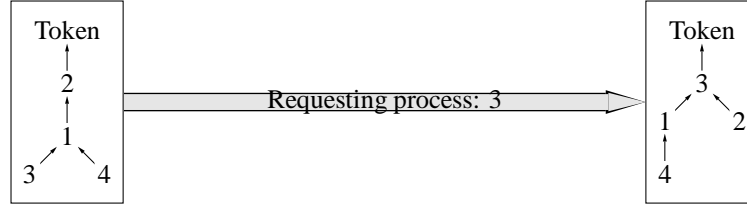


Figure 3. Abbreviated process “ascent”

Let \mathcal{A}_n be the set of abstract trees representing a n -processes system without requesting processes. Let α be an element of \mathcal{A}_n . Let us denote by $\mathcal{T}_i(\alpha)$ the result of the transformation of tree α when the process i requests the critical section. We note $\mathcal{T}_i^{-1}(\alpha) = \{\beta \in \mathcal{A}_n / \mathcal{T}_i(\beta) = \alpha\}$.

Figure 4 details \mathcal{A}_3 , the reachable state graph for three processes.

3.3. Complexity of the algorithm

In this subsection, we shall set out to compute the average number of messages needed for a process to perform a critical access in a system of n processes. Illustrative examples are given in the next subsections. The basic information for the complexity is the number of messages needed to satisfy a single request. This information can be extracted from the abstract tree structure.

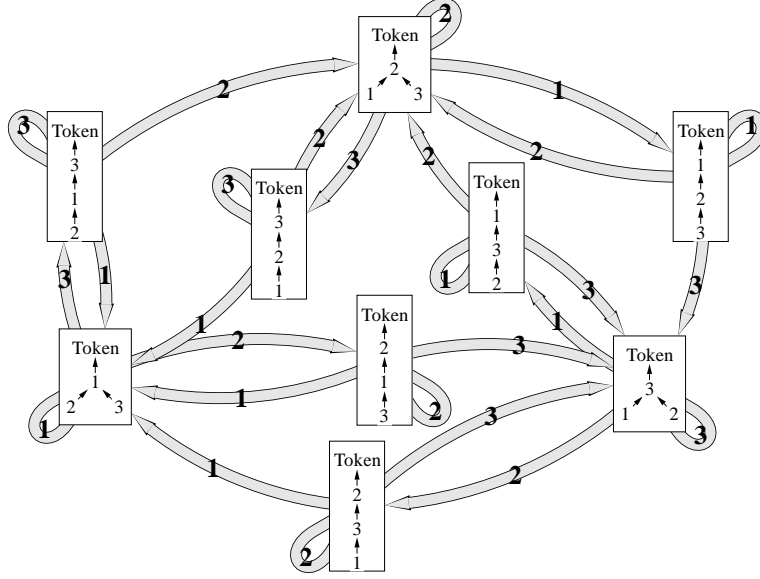


Figure 4. *Abbreviated reachable state graph for three processes*

The number of needed messages is

- zero if the requesting process holds the token,
- the height of the requesting process in the abstract tree when it asks: to each arrow from the requester to the root corresponds a message in the process ascent.

Let α be an element of \mathcal{A}_n . We note h_i^α the height of process i in the abstract tree α , and $holder(\alpha)$ the process holding the token in α .

Definition 1 (Average number of messages of a tree) Consider p_i the probability that the process i requests. The average number of messages for an abstract tree α is

$$\mathcal{M}_\alpha^n(p) = \sum_{i \in (\text{Sites} \setminus \{holder(\alpha)\})} p_i \cdot h_i^\alpha.$$

In the equiprobabilistic case (called EQUI) defined by $\forall i \in \text{Sites}, \text{EQUI}_i = \frac{1}{n}$, it gives $\mathcal{M}_\alpha^n(\text{EQUI}) = \frac{1}{n} \cdot \sum_{i \in (\text{Sites} \setminus \{holder(\alpha)\})} h_i^\alpha$.

We have to compute the “average” of the values $\mathcal{M}_\alpha^n(p)$ over the elements of \mathcal{A}_n . This “average” is first defined over trees reached after j requests. In a second time, we show that this average admits a limit when $j \rightarrow \infty$, which is used as a definition for the complexity.

Definition 2 (Tree probability) The tree probability π_j^α is the probability for the abstract tree α to appear after j requests. For the initial abstract tree a we have $\pi_1^a = 1$.

Otherwise, for any request distribution p , the tree probabilities are inductively defined by

$$\pi_{j+1}^\alpha = p_{holder(\alpha)} \sum_{\beta \in \mathcal{T}_{holder(\alpha)}^{-1}(\alpha)} \pi_j^\beta. \quad (1)$$

Equation 1 is a classical formula of Markov theory written in our specific case, where the last requester becomes the root of the resulting tree. This property can be expressed formally by

$$\forall \beta \in \mathcal{A}_n, holder(\mathcal{T}_i(\beta)) = i. \quad (2)$$

Definition 3 (Average number of messages) *The average number of messages $\mathcal{S}_j^n(p)$ sent by n processes, with request distribution p and after j requests, is defined by*

$$\mathcal{S}_j^n(p) = \sum_{\alpha \in \mathcal{A}_n} \pi_j^\alpha \cdot \mathcal{M}_\alpha^n(p). \quad (3)$$

When a request probability p_i is assigned to each process i , the algorithm becomes a Markov chain: each state is determined by a former state and the stochastic choice of a requesting site.

The strong connexity of the reachable state graph is proved in [8], Lemma 10. So, assuming that none of the p_i are 0 (which would be of no interest), this Markov chain is *irreducible*.

Moreover, it is *aperiodic*, since the probability of transition from any state α to itself is the positive value p_r , where r is the root of α .

For any irreducible and aperiodic finite Markov chain, one have a general result of convergence of the state probabilities to an invariant distribution (see, for instance, Theorem 1.8.3 in [9]). This theorem justifies the following definition for the complexity.

Definition 4 (Complexity) *The complexity $\mathcal{C}_n(p)$ of a n processes system is the limit of $\mathcal{S}_j^n(p)$ when j tends toward infinity:*

$$\mathcal{C}_n(p) = \lim_{j \rightarrow \infty} \mathcal{S}_j^n(p). \quad (4)$$

When the number j of requests tends toward ∞ , the probability π_j^α to reach the abstract tree α after j requests tends to a limit denoted π^α and the complexity can be computed by

$$\mathcal{C}_n(p) = \sum_{\alpha \in \mathcal{A}_n} \pi^\alpha \cdot \mathcal{M}_\alpha^n(p). \quad (5)$$

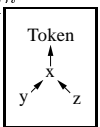
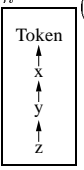
Theorem 1 ([5, 8]) *For equiprobabilistic distribution over the set of processes (representing the probability of asking the critical section) the complexity with the non-concurrent hypothesis is equal to H_{n-1} : $\mathcal{C}_n(\text{EQUI}) = H_{n-1}$.*

Recall that the harmonic number H_i is defined as $H_i = \sum_{n \in [1..i]} \frac{1}{n}$. It is known that $H_i \approx \log i + \gamma$, where γ is the Euler's constant which is around 0.577. This result has been obtained by two different manners in [5] and [8].

$$\pi \begin{array}{c} \boxed{\begin{array}{c} \text{Token} \\ \uparrow \\ \text{x} \\ \swarrow \quad \searrow \\ \text{y} \quad \text{z} \end{array}} = \frac{p_x}{2} \quad \text{and} \quad \pi \begin{array}{c} \boxed{\begin{array}{c} \text{Token} \\ \uparrow \\ \text{x} \\ \uparrow \\ \text{y} \\ \uparrow \\ \text{z} \end{array}} = \frac{p_x p_y}{2(1-p_x)} \end{array}$$

is a fixed point of these equations, as given in Figure 5, which exactly corresponds to Figure 4. More over, the sum of its coefficients is 1. Then, π is the tree probabilities limit needed to compute the complexity, together with the average numbers of messages given by:

$$\mathcal{M}^n(p) = 2(p_y + p_z) \quad \text{and} \quad \mathcal{M}^n(p) = 2p_y + 3p_z.$$

We can now explicit the complexity formula for 3 sites, using Equation 5:

$$\mathcal{C}_3(p_1, p_2, p_3) = 4(p_1p_2 + p_1p_3 + p_2p_3) + p_1p_2p_3 \left(\frac{1}{1-p_1} + \frac{1}{1-p_2} + \frac{1}{1-p_3} \right) \quad (8)$$

4. Non-equiprobabilistic case

4.1. Intuition and conjecture

As the average height of a process at the time it asks depends on how many of its “child” processes requests it has forwarded, it follows “the more a process asks, the higher it is in the abstract tree”. So, the basic intuition can be expressed: “*the more a process asks, the less it costs*”. To give a mathematical content to this abstract intuition, the key step is to say: “Between all distributions, equiprobability has the worst average-cost”.

Let $\mathcal{P}_n = \{p \in [0, 1]^{Sites} / \sum_{i \in Sites} p_i = 1\}$ be the set of probability distributions over the set of processes.

Conjecture 1 *For any probability distribution p over the set of processes (representing the probability of asking the critical section) the average complexity with the non-concurrent hypothesis is less than the average complexity of the equiprobabilistic distribution (H_{n-1}):*

$$\forall p \in \mathcal{P}_n, \mathcal{C}_n(p) \leq \mathcal{C}_n(\text{EQUI}) \leq H_{n-1}$$

The *limit state probability* π^α forms a vector π over the set \mathcal{A}_n of abstract trees, which is a fixed point for the application defined by the reduced state graph. As a consequence, π^α can - at least theoretically - be computed exactly by symbolic calculus. This strategy leads to the results of subsection 4.2. When the exact computations are too heavy to be performed, the fixed point can still be approached iteratively by a convergence method. The results of this second strategy are given in subsection 4.3.

4.2. Exact computation results

From an algebraic point of view, the limit tree probability vector π is an eigenvector of the Markovian transition matrix, associated to eigenvalue 1. Moreover, it is a probability distribution, so the sum of its coefficients is 1. It can be shown that these two properties characterize π and allow its formal computation as a rational function of p_1, \dots, p_n . In practice, we have got reasonable computation times only for small values of n .

4.2.1. Case of two sites

Let us prove the conjecture for two requiring sites. In this simple case, the reduced state graph has two nodes and four arrows, which are the two rooted trees with 2 nodes labeled 1 and 2.

The average number of messages for such a tree equals two times the probability of the non-root site. One can check that each rooted tree probability converges in one step to its final value, which is the probability of its root site.

So the complexity C_2 is given by:

$$C_2(p_1, p_2) = 4p_1p_2.$$

With $p_1 + p_2 = 1$, the complexity C_2 is obviously maximal for $p_1 = p_2 = \frac{1}{2}$ and its maximal value is $H_1 = 1$.

4.2.2. Case of three sites

Let us prove the conjecture for $n = 3$,

$$\forall (p_1, p_2, p_3) \in \mathcal{P}_3, C_3(p_1, p_2, p_3) \leq H_2$$

equivalent to the following inequation, remembering that $H_2 = 3/2$:

$$\begin{aligned} & (8(p_1p_2 + p_1p_3 + p_2p_3) - 3)(1 - p_1)(1 - p_2)(1 - p_3) \\ & + 2p_1p_2p_3((1 - p_2)(1 - p_3) + (1 - p_1)(1 - p_3) + (1 - p_1)(1 - p_2)) \leq 0. \end{aligned} \quad (9)$$

The value $p_2 = 1$ is of no interest: it would imply $p_1 = p_3 = 0$, trivial case of a single requesting site. For any fixed value of p_2 in $[0, 1[$, we define the function $f(p_1)$ as the left hand side of Inequation 9, when p_3 is replaced by $1 - p_1 - p_2$. The symmetry of $C_3(p_1, p_2, p_3)$ in p_1 and p_3 , $C_3(p_1, p_2, p_3) = C_3(p_3, p_2, p_1)$, implies that $f(p_1) = f(1 - p_1 - p_2)$, so that the function g defined by:

$$g(p_1) = 8f\left(p_1 + \frac{1 - p_2}{2}\right) \quad (10)$$

has even parity. In Definition 10, the multiplicative coefficient 8 is introduced for convenience only, in order to cancel all the fraction denominators in the following explicit expression of g :

$$\begin{aligned} g(p_1) &= Ap_1^4 + Bp_1^2 + g(0) \\ \text{with } A &= 16(4 - 3p_2), \\ B &= 8(-1 - 10p_2 + 12p_2^2 - 3p_2^3) \\ \text{and } g(0) &= (p_2 - 1)(p_2 + 2)(p_2 + 1)(-1 + 3p_2)^2. \end{aligned}$$

The conjecture is equivalent to the negativity of g over $[-\frac{1-p_2}{2}, \frac{1-p_2}{2}]$, that we prove by the study of the real zeros of g .

With $16(4 - 3p_2)p_1^4$ as leading term, $g(p_1)$ tends to $+\infty$ when p_1 tends toward $+\infty$ or $-\infty$. The value

$$g\left(\frac{1-p_2}{2}\right) = 8p_2(p_2 - 1)(8p_2^2 - 8p_2 + 3)$$

is strictly negative for any admissible value of p_2 . These two properties of g , polynomial of degree 4 in p_1 , imply — together with even parity — that g admits at least 2 real zeros.

In fact, if p_2 is not $\frac{1}{3}$, there are exactly 2 real zeros, since the product of the four g zeros is also the clearly negative product $\frac{g(0)}{A}$ of the two roots of equation

$$AX^2 + BX + g(0) = 0. \quad (11)$$

The positive real zero α of g is the square root of the positive solution of Equation 11. The negative one is $-\alpha$.

If $p_2 = \frac{1}{3}$, $g(0) = 0$, but the other root of Equation 11 is $\frac{14}{27}$, so that the other real zeros of g are $\alpha = \sqrt{\frac{14}{27}}$ and $-\sqrt{\frac{14}{27}}$.

In both cases, the function g stays negative in $[-\alpha, \alpha]$. But g is known to be negative at $-\frac{1-p_2}{2}$ and at $\frac{1-p_2}{2}$. Then, g stays negative in our interval of interest $[-\frac{1-p_2}{2}, \frac{1-p_2}{2}]$. This ends the proof of the conjecture for 3 processes.

For $n = 4$, the complexity formula is too large to be reported here and higher values of n are treated by numerical computations, as shown in the next section.

4.3. Experimental results and refined conjecture

We have developed a program that scans all possible discretized – monotonic – distributions p for a given step s . We choose them monotonic to avoid identical probabilities according to a process permutation. The final result (Figure 6) is a set of points $(\nu(p), \mathcal{C}_n(p))$, where $\nu(p)$ is the variability of probability distribution p . This representation has an interesting property: any dot is under a line connecting the two extreme dots $(0, H_{n-1})$ and $(\frac{n-1}{n^2}, 0)$, where $\frac{n-1}{n^2}$ is the variance of the distribution with only one process requesting — this case being of complexity zero —. This experimental feature allows us to propose a conjecture specifying the term of first order as given below.

Conjecture 2 *For any probability distribution p over the set of process, representing the probability of asking the critical section, the complexity, with the non-concurrent hypothesis is less than $H_{n-1} - \nu(p) \cdot \frac{n^2 H_{n-1}}{n-1}$, where $\nu(p)$ is the variability of the probability distribution.*

Proof 1 (Case of two processes) *Remembering that the $\{p_1, p_2\}$ distribution variability is $\nu(p_1, p_2) = \frac{2p_1^2 + 2p_2^2 - 1}{4}$. We have $C_2(p_1, p_2) = 1 - 4\nu(p_1, p_2) = H_1 - \nu(p) \cdot \frac{2^2 H_1}{2-1}$*

The conjecture is exactly verified for $n = 2$ and can be checked for $n = \{3, 4\}$. For $n \in \{5, 6\}$ the Markovian method gives no doubt about its validity (Cf figure 6).

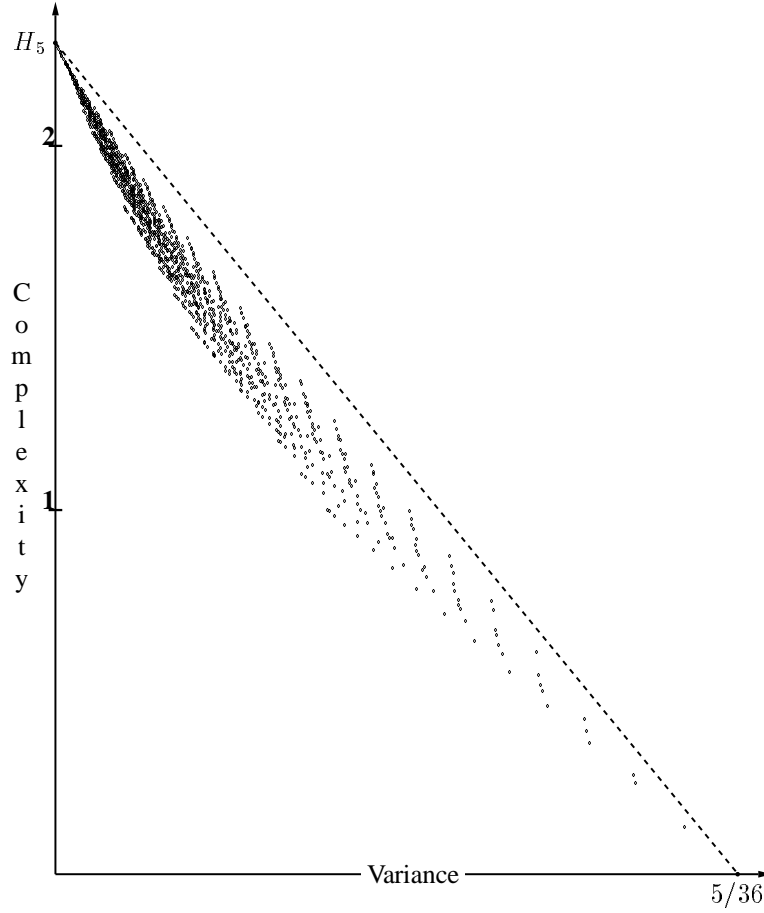


Figure 6. Six processes, probability distribution step $\frac{1}{30}$

5. Conclusion

The Naimi-Tréhel algorithm had already the particular advantage of being very efficient in the equiprobabilistic case. We have seen in this paper that it is more efficient in the non equiprobabilistic case. The intuition is “the more a process asks the less it costs”. One of the most known fact about this algorithm being that the complexity of a request is equal to the distance from the requesting process to the root, as each process on this path will forward its request, this is intuitively due to the fact that “the more a process asks, nearer it is from the root at the time it asks” because “fewer requests it will forward in the between”.

The first step performed in that article is to give a clear mathematical expression for that intuition: “equiprobability has to worst average case of all probability distri-

bution". We have developed a formal proof for a little number of process (2 and 3). When the system is larger, from 4 to 6, the computation of the fixed point by approximation gives no doubt about conjecture validity, even the refined one. For 7 processes or more, the only method which is not too large in computational resource is Monte-Carlo method which is too imprecise to assess the conjecture for the limit cases (around equiprobability).

We are now working in two directions. The first one is to prove our performance results for any number of processes. The second direction is an extension to concurrent requests, with a suitable complexity measure.

References

- [1] O. Carvalho and G. Roucairol. On mutual exclusion in computer networks. *Communications of the ACM*, 26(2):146–147, February 1983.
- [2] J. M. Hélary and A. Mostefaoui. A $O(\log_2 n)$ fault-tolerant distributed mutual exclusion algorithm based on open-cube structure. In *Proceedings of the 14th International Conference on Distributed Computing Systems*, pages 89–96, Poznan, Poland, June 1994. IEEE Computer Society Press.
- [3] J.-M. Hélary, A. Mostefaoui, and M. Raynal. General scheme for token-and tree-based distributed mutual exclusion algorithms. *IEEE Transactions on Parallel and Distributed Systems*, 5(11):1185–1196, November 1994.
- [4] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.
- [5] C. Lavault. Analysis of an efficient distributed algorithm for mutual exclusion (average-case analysis of path reversal). In *VAPP: CONPAR 90–VAPP IV: Joint International Conference on Vector and Parallel Processing*. LNCS 457, Springer-Verlag, 1992.
- [6] Mamoru Maekawa. A \sqrt{N} algorithm for mutual exclusion in decentralized systems. *ACM Transactions on Computer Systems*, 3(2):145–159, May 1985.
- [7] M. Naimi and M. Trehel. A distributed algorithm for mutual exclusion based on data structures and fault tolerance. In *6th International Phoenix Conference on computers and communications*, pages 35–39, Washington, D.C., USA, February 1987. IEEE Computer Society Press.
- [8] Mohamed Naimi, Michel Trehel, and Andre Arnold. A $\log(N)$ distributed mutual exclusion algorithm based on path reversal. *Journal of Parallel and Distributed Computing*, 34(1):1–13, April 1996.
- [9] J. R. Norris. *Markov Chains*. Cambridge University Press, 1997.
- [10] Kerry Raymond. A tree-based algorithm for distributed mutual exclusion. *ACM Transactions on Computer Systems*, 7(1):61–77, February 1989.
- [11] M. Raynal. A simple taxonomy for distributed mutual exclusion algorithms. *ACM Operating Systems Review, SIGOPS*, 25(2):47–50, April 1991.
- [12] Glenn Ricart and Ashok K. Agrawala. An optimal algorithm for mutual exclusion in computer networks. *Communications of the ACM*, 24(1):9–17, January 1981.