Chapitre 2

Algorithmes distribués probabilistes

2.1 Définitions

Un *algorithme probabiliste* est un algorithme qui utilise des tirages de type pile ou face ou des générateurs de nombres aléatoires. À la différence des algorithmes *déterministes*, le *hasard* joue un rôle fondamental dans l'exécution de tels algorithmes. Les algorithmes probabilistes permettent de fournir des solutions parfois plus "efficaces" que les solutions déterministes, ou encore des solutions pour des problèmes qui n'admettent pas de solution déterministe.

Plus formellement, voir [Tel00] chapitre 9, un *processus probabiliste*, est défini par un quadruple $p = (\mathcal{E}, \mathcal{J}, \rightarrow^0, \rightarrow^1)$ où

- $\mathcal{E}$ est l'ensemble d'états du processus,
- $\mathcal{J} \subset \mathcal{E}$ est un seous ensemble d'états initiaux,
- $\rightarrow^0$ et $\rightarrow^1$ sont des relations binaires sur $\mathcal{E}$ telles que le $i^{\text{ème}}$ pas du processus est exécuté selon la transition dans $\rightarrow^0$ si le résultat du tirage est 0 et selon la transition dans $\rightarrow^1$ sinon.

Un *algorithme probabiliste distribué* $\mathcal{A}_p$ est par conséquent une collection d'algorithmes locaux $\mathcal{A}_p = (\mathcal{A}_i)_{1 \leq i \leq n}$ telle que chaque algorithme $\mathcal{A}_i$ est un algorithme (ou processus) probabiliste.

2.2 Analyse des algorithmes distribués probabilistes

Les deux principales mesures auxquelles on s'intéresse sont le temps d'exécution de l'algorithme et le nombre ou la taille des messages échangés. On s'intéresse principalement à l'espérance mathématique de la valeur de la mesure considérée.

2.2.1 Complexité en temps

Dans ce chapitre, nous nous plaçons dans le cadre d'un système distribué synchrone. L'analyse de la complexité en temps des algorithmes proposés utilise l'hypothèse de synchronisme. C'est une hypothèse forte, néanmoins, comme c'est indiqué dans [Lav95], page

53, ceci permet d'obtenir une "bonne" estimation du temps d'exécution de l'algorithme si le système est supposé être asynchrone.

Complexité moyenne

On considère un système distribué synchrone $\mathcal{S}$ à $n \geq 1$ entités. Soit $\mathcal{T}$ une tâche à réaliser sur $\mathcal{S}$ et $\mathcal{A}_p = (\mathcal{A}_i)_{1 \leq i \leq n}$ un algorithme probabiliste permettant de réaliser la tâche $\mathcal{T}$. Sans perte de généralité, on considère que l'algorithme $\mathcal{A}_p$ démarre à l'instant $t = 0$. Nous définissons la v.a. T comme le nombre d'unités de temps qui s'écoulent entre l'instant $t = 0$ et la plus petite valeur de t telle que $\forall i \in \{1, 2, \cdots, n\}$, l'algorithme $\mathcal{A}_i$ est terminé à l'instant t. La complexité moyenne de l'algorithme $\mathcal{A}_p$ est donc l'espérance mathématique de la v.a. T.

Dans l'analyse des algorithmes probabilistes, on s'intéresse principalement à l'espérance de T. Cependant, une analyse plus fine, et parfois plus complexe, permet de calculer, du moins asymptotiquement, la distribution de T.

Dans beaucoup de cas, nous pouvons montrer que la probabilité que T dépasse $C(n)$ (une fonction de la taille n du graphe) est majorée par $1/n^{\alpha}$, avec $\alpha \geq 1$. On dit alors que $T \leq C(n)$ avec forte probabilité.

2.3 Las Vegas et Monte Carlo

Algorithmes de type Las Vegas

Un algorithme probabiliste $\mathcal{A}$ est de type *Las Vegas* si $\mathcal{A}$ fournit *toujours* une solution *correcte* mais dont la complexité en temps est une v.a. T.

Algorithmes de type Monte Carlo

Un algorithme probabiliste $\mathcal{A}$ est de type *Monte Carlo* si $\mathcal{A}$ retourne une solution correcte avec une probabilité non nulle.

2.3.1 Exemple : un algorithme simple d'élection dans K_n

Dans cette section, nous allons illustrer quelques unes des notions et notations introduites ci-dessus. Rappelons que le problème de l'élection consiste à choisir un élément et un seul dans un ensemble. Dans cet exemple, nous présentons et analysons un algorithme distribué probabiliste pour élir un sommet dans un graphe. L'algorithme est étudié pour le cas où le graphe est complet. Le fait de se restreindre au cas du graphe complet permet de ne pas se soucier du problème de la détection de la terminaison.

L'algorithme étudié est simple. Il est paramétré par un entier $N \geq 1$. Chaque sommet v tire uniformément un nombre $r(v)$ dans l'ensemble $\{1, 2, \cdots, N\}$, v envoie $r(v)$ à tous ses voisins et reçoit $r(u)$ de tous ses voisins. Le sommet v compare ensuite $r(v)$ avec les nombres tirés par ses voisins. Si v a tiré la plus grande valeur alors v est élu, sinon, il est battu. L'étude de l'algorithme consiste à étudier la probabilité pour un sommet v d'être élu ensuite, la probabilité qu'un sommet du graphe soit élu.

Algorithme 1 : *ElectionProba*$_1$(), élection dans un graphe complet.

v tire uniformément un élement $r(v)$ de l'ensemble $\{1, 2, \cdots, N\}$;
v envoie $r(v)$ à tous ses voisins;
v reçoit $r(u)$ de chaque voisin u;
si $r(v) > r(u)$ *pour tout sommet* u *de* V **alors**
  $etat_v := Elu$;
sinon
  $etat_v := NonElu$;
v envoie $etat_v$ à tous ses voisins;
v reçoit $etat_u$ de chaque voisin u;

Probabilité pour un sommet d'être élu

Nous avons :

Proposition 2.1 *Soit v un sommet quelconque de K_n. Soit $\mathcal{E}(v)$ l'événement v est élu. Alors :*

$$\mathbb{P}\left(\mathcal{E}(v)\right) = \frac{1}{N}\sum_{i=2}^{N}\left(\frac{i-1}{N}\right)^{n-1}. \tag{2.1}$$

Preuve. Soit v un sommet de K_n. L'événement $\mathcal{E}(v)$ a lieu si, et seulement si, la valeur tirée par v est plus grande que les valeurs tirées par tous les autres sommets de K_n. Si $r(v) = 1$, il est clair que v ne peut pas être élu. Si $r(v) = 2$, alors v est élu si, et seulement si, tous les autres sommets ont tiré la valeur 1. Da manière générale, si $r(v) = i \in \{2, 3, \cdots, N\}$, alors v est élu si, et seulement si, tous les autres sommets ont tiré une valeur $j < i$. Pour tout $i \in \{2, 3, \cdots, N\}$, notons $A_i(v)$ l'événement "v a tiré la valeur i" et $A_{<i}(v)$ l'événement "v a tiré une valeur $j < i$". L'évenement $\mathcal{E}(v)$ se réécrit :

$$\mathcal{E}(v) = \bigcup_{i=2}^{N}\left[A_i(v) \cap \bigcap_{u\neq v} A_{<i}(u)\right]. \tag{2.2}$$

On en déduit que :

$$\mathbb{P}\left(\mathcal{E}(v)\right) = \sum_{i=2}^{N}\left(\mathbb{P}\left(A_i(v)\right) \times \prod_{u\neq v}\mathbb{P}\left(A_{<i}(u)\right)\right). \tag{2.3}$$

Par ailleurs, tous les nombres de l'ensemble $\{1, 2, \cdots, N\}$ ont la même probabilité d'être tirés par v. Il en résulte que pour tout $i \in \{1, 2, \cdots, N\}$:

$$\mathbb{P}\left(A_i(v)\right) = \frac{1}{N} \text{ et } \mathbb{P}\left(A_{<i}(u)\right) = \frac{i-1}{N}, \tag{2.4}$$

La relation (2.3) devient alors :

$$\begin{aligned}\mathbb{P}\left(\mathcal{E}(v)\right) &= \sum_{i=2}^{N}\left(\frac{1}{N} \times \prod_{u\neq v}\frac{i-1}{N}\right)\\ &= \frac{1}{N}\sum_{i=2}^{N}\left(\frac{i-1}{N}\right)^{n-1}.\end{aligned} \tag{2.5}$$

Ce qui termine la preuve.

Le paramètre N est fondamental dans l'analyse de l'algorithme. En effet :

- Si $N = 1$, l'algorithme échoue et aucun sommet ne peut être élu.
- Si $N = 2$, cela revient à faire des tirages dans l'ensemble $\{0, 1\}$. Un sommet v est donc élu si, et seulement si, il est le seul à avoir tiré 1. L'expression (2.1) devient alors :
$$\mathbb{P}(\mathcal{E}(v)) = \frac{1}{2^n}.$$
- Si $N \to \infty$, nous avons :
$$\begin{aligned}\sum_{i=2}^{N}(i-1)^{n-1} &= \sum_{i=1}^{N-1} i^{n-1} \\ &\sim \int_1^{N-1} x^{n-1}dx \\ &= \frac{(N-1)^n - 1}{n}.\end{aligned}$$
D'où
$$\begin{aligned}\mathbb{P}(\mathcal{E}(v)) &\sim \frac{1}{n}\left((1-\frac{1}{N})^n - \frac{1}{N^n}\right) \\ &\to \frac{1}{n}, \text{ quand } N \to \infty.\end{aligned}$$
- Si $N = n$, un raisonnement similaire aboutit à :
$$\begin{aligned}\mathbb{P}(\mathcal{E}(v)) &\sim \frac{1}{n}\left((1-\frac{1}{n})^n - \frac{1}{n^n}\right), \text{ quand } n \to \infty \\ &\sim \frac{1}{ne} - \frac{1}{n^{n+1}}.\end{aligned}$$

Probabilité d'aboutir à une élection

Soit $\mathcal{E}$ l'évenement *Il y a un sommet élu dans le graphe*. Il est alors clair que :

$$\mathcal{E} = \{\exists v \in V \text{ tel que } \mathcal{E}(v)\}.$$

Par ailleurs, pour tous $u, v \in V$, $\mathcal{E}(u) \cap \mathcal{E}(v) = \emptyset$. Il en résulte que :

$$\begin{aligned}\mathbb{P}(\mathcal{E}) &= \mathbb{P}(\{\exists v \in V \text{ tel que } \mathcal{E}(v)\}) \\ &= \sum_{v \in V} \mathbb{P}(\mathcal{E}(v)) \\ &= \frac{n}{N}\sum_{i=2}^{N}\left(\frac{i-1}{N}\right)^n.\end{aligned}$$

Si $N = n$, il résulte de ces expressions que :

$$\begin{aligned}\mathbb{P}(\mathcal{E}) &\sim \frac{1}{e} - \frac{1}{n^n}, \text{ quand } n \to \infty. \\ &\sim \frac{1}{e} \geq 1/3.\end{aligned}$$

L'algorithme est par conséquent un algorithme de type Monte Carlo.

La complexité en temps de l'algorithme $ElectionProba_1()$ est 1. Cependant, sauf pour $N \to \infty$, la probabilité que $ElectionProba_1()$ échoue est strictement suppérieure à 0. Pour $N = n$, nous avons vu que $ElectionProba_1()$ est de type Monte Carlo. Nous avons également le résultat suivant :

Si $N = n^2$, alors $ElectionProba_1()$ réalise une élection avec forte probabilité.

En effet, si on remplace N par n^2 dans (2.6), nous obtenons

$$\begin{aligned} \mathbb{P}(\mathcal{E}) &\sim \left(1 - \frac{1}{n^2}\right)^n, \text{ quand } n \to \infty \\ &\sim e^{-1/n} \\ &> 1 - \frac{1}{n}. \end{aligned}$$

On en déduit que la probabilité que l'algorithme $ElectionProba_1()$ réalise une élection est suppérieure à $1 - \frac{1}{n}$. Autrement dit, l'algorithme réussit avec forte probabilité.

On considère à présent l'algorithme suivant :

Algorithme 2 : L'algorithme $ElectionProba_2()$.

répéter
  $ElectionProba_1()$;
jusqu'à $etat_v = Elu$ *ou* $\exists u$ *tel que* $etat_u = Elu$;

L'algorithme $ElectionProba_2()$ ne s'arrête que si un sommet du graphe est élu. Donc si $ElectionProba_2()$ termine, alors le résultat est correct. Il reste donc à étudier sa complexité en temps (pour différentes valeurs de N). Cette complexité est une v.a. T.

Pour cela, nous définissons le nouvel événement :

$$\mathcal{E}_i = \{\text{il y a une élection à la } i^{\text{ème}} \text{ itération, }\} \text{ pour tout } i \geq 1.$$

Il est alors clair que :

$$\text{Pour tout } i \geq 1,\ T = i \text{ s, et seulement si, } \mathcal{E}_i.$$

L'événement $\mathcal{E}_i$ correspond à la situation suivante : pendant les $(i-1)^{\text{èmes}}$ itérations de l'algorithme $ElectionProba_1()$, aucun sommet n'a été élu, et à la $i^{\text{ème}}$ exécution de $ElectionProba_1()$, un sommet a été élu. Si on interprète le fait d'obtenir un sommet élu comme un *succès* et l'événement inverse comme un *échec*, la v.a. T correspond au nombre d'itérations avant d'obtenir un succès. La v.a. T est par conséquent une v.a. suivant une loi géométrique de paramètre $p = \mathbb{P}(\mathcal{E})$. Autrement dit :

$$\mathbb{P}(T = i) = (1-p)^{i-1} p.$$

Nous en déduisons :

– Si $N = 2$,

$$\mathbb{P}(T = i) = \left(1 - \frac{1}{2^n}\right)^{i-1} \frac{1}{2^n} \text{ et } \mathbb{E}(T) = \frac{1}{p} = 2^n.$$

– Si $N = n$,

$$\mathbb{P}(T = i) = \left(1 - \frac{1}{e}\right)^{i-1} \frac{1}{e},\ \mathbb{E}(T) = \frac{1}{p} = e \text{ et } \mathbb{V}ar(T) = e - 1.$$

L'espérance de la v.a. est une constante et sa variance également. On cherche à mesurer l'écart de la v.a. T de son espérance e. Une première tentative consiste à utiliser l'inégalité de Markov (voir (**??**)) :

$$\mathbb{P}(T > k) \leq \frac{\mathbb{E}(T)}{k} = \frac{e}{k}.$$

Si $k = e$, l'inégalité devient :

$$\mathbb{P}(T > e) \leq \frac{\mathbb{E}(T)}{e} = 1.$$

Ce qui est une majoration triviale.
De même, si on utilise l'inégalité de Bienaymé-Tchebicheff (voir (**??**)), nous avons :

$$\mathbb{P}(\mid T - \mathbb{E}(T) \mid \geq k) \leq \frac{\mathbb{V}ar(T)}{t^2} = \frac{e-1}{k^2}.$$

Si $k = e$, l'inégalité devient :

$$\mathbb{P}(\mid T - e \mid \geq e) \leq \frac{e-1}{e^2} \leq 0,2325....$$

Ce qui donne une majoration non triviale mais insuffisante. Cependant, il est clair que pour tout $k \geq 1$,

$$\begin{aligned}
\mathbb{P}(T \geq k) &= \frac{1}{e} \sum_{i \geq k} \left(1 - \frac{1}{e}\right)^{i-1} \\
&= \frac{1}{e} \sum_{j \geq O} \left(1 - \frac{1}{e}\right)^{j+k-1} \\
&= \frac{1}{e} \left(1 - \frac{1}{e}\right)^{k-1} \sum_{j \geq O} \left(1 - \frac{1}{e}\right)^{j} \\
&= \frac{1}{e} \left(1 - \frac{1}{e}\right)^{k-1} \frac{1}{1 - \left(1 - \frac{1}{e}\right)} \\
&= \left(1 - \frac{1}{e}\right)^{k-1}
\end{aligned}$$

Il en résulte que pour tout $k \geq e \log n + 1$,

$$\mathbb{P}(T \geq k) \leq \frac{1}{n}.$$

La complexité de l'algorithme $ElectionProba_2()$ est ainsi en $O(\log n)$ avec forte probabilité.

Bibliographie

[Lav95] Ch. Lavault. *Évaluation des algorithmes distribués*. Hermes, 1995.

[Tel00] G. Tel. *Introduction to distributed algorithms*. Cambridge University Press, 2000.