Complexité de Communication

Extrait d'un chapitre écrit par :
Iordanis Kerenidis, Frédéric Magniez et David Xiao
Adapté par Samuel Guerin

23 novembre 2014

Résumé

Imaginons un scénario où deux protagonistes (ou joueurs), Alice et Bob, souhaitent calculer de manière collaborative une fonction $f(x, y)$ alors que x est détenue par Alice et y par Bob. La complexité de communication de la fonction f est le nombre de bits qu'Alice et Bob doivent s'échanger afin de pouvoir calculer $f(x, y)$. Les autres ressources, telles que temps et espace mémoire, sont ici ignorées afin de se focaliser uniquement sur la communication requise.

Plus formellement, Alice et Bob s'échangent des bits à tour de rôle en suivant un protocole, qui définit à chacun instant qui doit envoyer le prochain bit, ou si le protocole est terminé. Lorsque le protocole est terminé, la valeur de $f(x, y)$ doit alors être connue des deux joueurs, ou bien d'un seul des deux joueurs dans le contexte de communication unilatérale (où seule Alice envoie un message à Bob). Ce scénario modélise une communication parfaite, où chaque message arrive à son destinataire sans être perdu ou modifié. Ce sera le cas de tout ce chapitre.

L'étude de la complexité de communication permet d'unifier des preuves de bornes inférieures dans plusieurs domaines dont les automates, les machines de Turing, les circuits VLSI, les arbres de décision, les branching programs. Nous appliquerons quand à nous ces résultats aux algorithmes de streaming

Toutes les notions et résultats présentés se généralisent aux relations, mais nous ne considèrerons que les fonctions.

Nous regroupons ci-dessous les principales fonctions dont la complexité de communication sera étudiée. Notons dorénavant $[n] = \{1, 2, \ldots, n\}$ pour tout entier n.

$$
\begin{aligned}
\textsc{Index}_n : \quad & \{0,1\}^n \times [n] \to \{0,1\} \\
& (x, i) \mapsto x_i. \\
\textsc{Equality}_n : \quad & \{0,1\}^n \times \{0,1\}^n \to \{0,1\} \\
& (x, y) \mapsto \begin{cases} 1, & \text{si } x = y\,; \\ 0, & \text{sinon.} \end{cases} \\
\textsc{Disjointness}_n : \quad & \{0,1\}^n \times \{0,1\}^n \to \{0,1\} \\
& (x, y) \mapsto \begin{cases} 1, & \text{si } x_i = y_i = 1 \text{ pour un } i \in [n]\,; \\ 0, & \text{sinon.} \end{cases}
\end{aligned}
$$

Enfin, dans tout ce chapitre les logarithmes seront pris en base 2.

1 Généralités

1.1 Cas déterministe

Lorsque l'intéraction entre les deux joueurs est permise, la sortie du protocole doit pouvoir être calculée par chacun des deux joueurs. Un protocole est alors défini comme suit.

Définition 1.1 (Protocole déterministe (cas général))**.** *Soient X, Y deux ensembles finis. Un* protocole déterministe *π entre deux joueurs, Alice (A) et Bob (B), est la description d'une conversation entre Alice et Bob comme suit. Pour toute entrée $x \in X$ d'Alice et entrée $y \in Y$ de Bob :*

1. *Alice envoie un premier message $m_1(x)$ de ℓ_1 bits à Bob ;*
2. *Bob répond par un deuxième message $m_2(y, m_1)$ de $\ell_2(m_1)$ bits à Alice ;*
3. *Et ainsi de suite tant que le protocole n Pour $i \geq 1$, Alice envoie $m_{2i-1}(x, m_1, m_2, \ldots, m_{2i-2})$ de $\ell_{2i-1}(m_1, m_2, \ldots, m_{2i-2})$ bits, et Bob $m_{2i}(y, m_1, m_2, \ldots, m_{2i-1})$ de $\ell_{2i}(m_1, m_2, \ldots, m_{2i-1})$ bits.*
4. *La fin du protocole est décidée simultanément par les deux joueurs en fonction des messages envoyées $m_1, m_2, \ldots$ ainsi que de x pour Alice, et de y pour Bob.*
5. *A la fin du protocole, Alice et Bob peuvent calculer une même* sortie *$out(x, y)$, qui dépend des messages échangés ainsi que de x pour Alice, et de y pour Bob.*

La complexité de communication $\mathsf{C}(\pi, x)$ de π sur l'entrée x est la taille en bits du message $m(x)$ d'Alice. La complexité de communication $\mathsf{C}(\pi)$ de π est la valeur maximale de sa complexité de communication pour toutes les entrées $x \in X$. Remarquons que la taille ℓ du message d'Alice est décidée par le protocole, et non pas par l'entrée x d'Alice. En particulier, des scénarios où Bob pourrait tirer de l'information de la "non-réception" d'un message d'Alice, ou encore de la taille du message d'Alice, ne sont pas autorisés. On défini alors la notion de compléxité de la communication par :

$$\mathsf{D}(f) = \min\{\mathsf{C}(\pi) : \textit{protocole } \pi \textit{ déterministe calculant } f\}.$$

Pour toute fonction $f : X \times Y \to Z$, on a déjà que

$$\mathsf{D}(f) \leq \log(max(|X|, |Y|)) + \log |Z|,$$

en considérant le protocole qui envoie la valeur de x, ou de y, à l'autre joueur. Celui ci calculant alors la valeur de la fonction associée, qu'il renvoie au premier joueur.

1.2 Cas probabiliste

Lorsque les joueurs, Alice et Bob, ont accès à une source aléatoire, le protocole devient probabiliste : son exécution, sa transcription et sa sortie peuvent dépendre des choix aléatoires d'Alice et Bob. La notion de calcul d'une fonction f par un protocole est alors relâchée comme pour les algorithmes en tolérant probabilité d'erreur au plus ε, pour $0 < \varepsilon < 1/2$, c'est-à-dire en imposant que la sortie du protocole soit correcte avec probabilité au moins $(1 - \varepsilon)$, pour chaque entrée (x, y). Cette probabilité d'erreur, pour une entrée (x, y) fixée, ne dépend donc que des choix aléatoires réalisés par les joueurs.

La source aléatoire peut être de deux natures différentes :

— *Soit elle est propre à chaque joueur, c'est-à-dire les bits aléatoires r_A d'Alice sont indépendants des bits aléatoires r_B de Bob. On parle alors d'*aléa privé *;*

— *Soit elle est commune aux deux joueurs, c'est-à-dire que les bits aléatoires d'Alice sont identiques à ceux de Bob ($r_A = r_B$). On parle alors d'*aléa public.

Etant donnée une fonction f, les notions de complexités des protocoles probabilistes sont alors définies comme suit :

— *Complexité de communication probabiliste à aléa privé :*

$$\mathsf{R}_\varepsilon(f) = \min\left\{\mathsf{C}(\pi) : \begin{array}{l} \textit{protocole probabiliste } \pi \textit{ calculant } f \\ \textit{avec aléa privé et erreur bornée } \varepsilon \end{array}\right\};$$

— *Complexité de communication probabiliste à aléa public :*

$$\mathsf{R}^{\mathsf{pub}}_\varepsilon(f) = \min\left\{\mathsf{C}(\pi) : \begin{array}{l} \textit{protocole probabiliste } \pi \textit{ calculant } f \\ \textit{avec aléa privé } \textbf{\textit{et}} \textit{ public, et erreur bornée } \varepsilon \end{array}\right\}.$$

Disposant d'un aléa publique, les joueurs peuvent décider de ne garder chacun que un bit aléatoire sur deux. Ils disposent alors d'une source aléatoire privée. De même il peuvent décider d'ignorer la source. On a donc :

$$\mathsf{R}^{\mathsf{pub}}_\varepsilon(f) \le \mathsf{R}_\varepsilon(f) \le \mathsf{D}(f).$$

Exemple 1.2. *Alors que* $\mathsf{D}(\textsc{Equality}_n) \le n+1$ *et on vera plus tard que* $\mathsf{D}(\textsc{Equality}_n) = n+1$*, les protocoles 1 et 2 permettent d'établir que*

$$\mathsf{R}(\textsc{Equality}_n) \le \lceil \log(n/\varepsilon) \rceil + 1 \quad \textit{et} \quad \mathsf{R}^{\mathsf{pub}}_\varepsilon(\textsc{Equality}_n) \le \lceil \log(1/\varepsilon) \rceil.$$

Ces protocoles renvoient toujours '$x = y$' lorsque $x = y$. Leurs complexités vérifient l'inégalité ci-dessus. Lorsque $x \neq y$, le protocole 1 se trompe avec probabilité au plus ε car le polynôme $P = \sum_{i=1}^{n}(x_i - y_i)a^{n-i}$ s'annule alors en au plus n points (car lorsque p est premier, l'anneau $\mathbb{Z}_p$ est un corps), et que a est pris au hasard dans un ensemble de taille au moins n/ε. Pour le protocole 2, observons que si $w \in \{0,1\}^n$ et $w \neq 0^n$, alors $\bigoplus_{i=1}^{n} w_i r_i = 0$ avec probabilité $1/2$, lorsque r est choisi uniformément au hasard dans $\{0,1\}^n$. Par conséquent, $w := (x_i \oplus y_i)_{i=1,2,\ldots,n}$ satisfait $w \neq 0^n$ lorsque $x \neq y$. De plus $u = v$ si et seulement si $\bigoplus_{i=1}^{n} w_i r_i^t = 0$, pour $t = 1, 2, \ldots, k$. Donc le protocole 2 se trompe avec probabilité $1/2^k \le \varepsilon$.

Protocole 1 :

- Alice :

— *Choisir un nombre premier p quelconque tel que $n/\varepsilon \le p \le 2n/\varepsilon$*

— *Choisir uniformément au hasard un entier a tel que $0 \le a < p$*

— *Calculer $u = \sum_{i=1}^{n} x_i a^{n-i} \mod p$*

— *Envoyer $m_1 := (p, a, u)$ à Bob*

- Bob :

— *Calculer $v = \sum_{i=1}^{n} y_i a^{n-i} \mod p$*

— *Si $u = v$, stopper et déclarer '$x = y$'*

— *Sinon stopper et déclarer '$x \neq y$'*

Protocole 2 :

- *Alice et Bob :*
  - *Choisir* $k := \lceil 1/\varepsilon \rceil$ *mots* $r^1, r^2, \ldots, r^k$ *de* n *bits uniformément au hasard*
- *Alice :*
  - *Calculer* $u_t = \bigoplus_{i=1}^n x_i r_i^t$, *pour* $t = 1, 2, \ldots, k$
  - *Envoyer* $m_1 := u$ *à Bob'*
- *Bob :*
  - *Calculer* $v_t = \bigoplus_{i=1}^n y_i r_i^t$, *pour* $t = 1, 2, \ldots, k$
  - *Si* $u = v$, *stopper et déclarer '*$x = y$*'*
  - *Sinon stopper et déclarer '*$x \neq y$*'*

2 Communication unilatérale

Définition 2.1 (Protocole déterministe unilatéral)**.** *Soient* X, Y *deux ensembles finis. Un* protocole déterministe unilatéral π *entre deux joueurs, Alice et Bob, est la description d'une conversation unilatérale d'Alice vers Bob comme suit. Pour toute entrée* $x \in X$ *d'Alice et entrée* $y \in Y$ *de Bob :*

1. *Alice envoie un message* $m(x)$ *de* ℓ *bits à Bob, où* $m : X \to \{0,1\}^\ell$ *;*
2. *Bob calcule la* sortie $out(m, y)$*, où* out *est définie sur* $\{0,1\}^\ell \times Y$*.*

Puisque m *est définie par l'entrée* x*, la sortie* out *du protocole est donc aussi une fonction de* (x, y) *qu'on pourra noter* $out(x, y)$ *au lieu de* $out(m, y)$*.*

Comme pour les algorithmes, un protocole π calcule *une fonction* f *si* $out(x, y) = f(x, y)$ *pour toutes entrées* (x, y)*. Etant donnée une fonction* f*, définissions la* complexité de communication déterministe unilatérale *par la quantité*

$$\overrightarrow{\mathsf{D}}(f) = \min\{\mathsf{C}(\pi) : \textit{protocole } \pi \textit{ déterministe unilatéral calculant } f\}.$$

Puisqu'Alice peut toujours se contenter d'envoyer la totalité de son entrée x *à Bob, on peut déjà établir que*

$$\overrightarrow{\mathsf{D}}(f) \leq \lceil \log |X| \rceil.$$

Comme nous le voyons dans l'exemple précédent, $\overrightarrow{\mathsf{D}}(f)$ *peut être beaucoup plus petite que le majorant* $\lceil \log |X| \rceil$*. Afin de caractériser* $\overrightarrow{\mathsf{D}}(f)$*, introduisons la notion de* matrice de communication M_f *de* f*. Cette matrice est indexée par* $x \in X$*, pour les lignes, et par* $y \in Y$*, pour les colonnes, et satisfait*

$$(M_f)_{x,y} = f(x, y).$$

Théorème 2.2. *Soient* X, Y *deux ensembles finis, et* f *une fonction définie sur* $X \times Y$*.*

$$\overrightarrow{\mathsf{D}}(f) = \lceil \log(\textit{nombre de lignes distinctes dans } M_f) \rceil.$$

Démonstration. Posons $k =$ nombre de lignes distinctes dans M_f, et considérons un protocole π quelconque pour f. Nous commençons par montrer qu'Alice doit pouvoir envoyer au moins k messages différents lorsque x varie, c'est-à-dire que $\mathsf{C}(\pi) \geq \lceil \log(k) \rceil$. Pour cela, montrons que si deux lignes x et x' de M_f sont distinctes alors nécessairement $m(x) \neq m(x')$. Par contraposition, si $m(x) = m(x')$ alors pour toute entrée $y \in Y$ le protocole calcule $out(m(x), y) = out(m(x'), y)$.

Protocole 1 – Protocole unilatéral optimal

- Préparation :
  Soit $L_1, L_2, \ldots, L_k$ les lignes sans multiplicité de M_f
  ordonnées arbitrairement (ordre connu d'Alice et Bob)
- Alice :
  Soit $i \in [k]$ l'entier tel que $L_i = (M_f(x,y))_{y \in Y}$
  Envoyer $m := i$ à Bob
- Bob : Calculer $out := L_m[y]$

Puisque π calcule f, il s'en suit que $f(x,y) = f(x',y)$, pour tout $y \in Y$, et donc que les lignes x et x' de M_f sont identiques.

Pour montrer $\overrightarrow{\mathsf{D}}(f) \leq \lceil \log k \rceil$, nous construisons un protocole π pour f tel que $\mathsf{C}(\pi) = \lceil \log k \rceil$. Soit L l'ensemble des lignes de M_f. Par hypothèse, l'ensemble L est de taille k. Le protocole 1 calcule alors correctement f et est de complexité annoncée. □

Une conséquence de ce résultat est la caractérisation des complexités suivantes, dont nous laissons les preuves en exercice :

Lemme 2.3.
$$\overrightarrow{\mathsf{D}}(\textsc{Equality}_n) = \overrightarrow{\mathsf{D}}(\textsc{Index}_n) = \overrightarrow{\mathsf{D}}(\textsc{Disjointness}_n) = n.$$

3 Communication bilatérale déterministe

La transcription $m(x,y)$ *de* π *sur l'entrée* (x,y) *est la concaténation des messages échangés* $m(x,y) := m_1 m_2 \ldots$. *La* complexité (de communication) $\mathsf{C}(\pi, x, y)$ *de* π *sur l'entrée* (x,y) *est la taille en bits de la transcription correspondante. La* complexité (de communication) $\mathsf{C}(\pi)$ *de* π *est la valeur maximale de sa complexité de communication pour toutes les entrées* $(x,y) \in X \times Y$.

Une propriété fondamentale des protocoles de communication est que les entrées (x,y) menant à la même transcription forment un rectangle : Un *rectangle* est un ensemble R tel que pour tout $(x,y) \in R$ et $(x',y') \in R$, alors $(x,y') \in R$ et $(x',y) \in R$. Un corollaire est que la sortie d'un protocole ne dépend en fait que de sa transcription. Il est donc possible à quiconque qui écoute la communication entre Alice et Bob de calculer la fonction sans même connaître x ou y !

Lemme 3.1. *Soit* π *un protocole déterministe sur* $X \times Y$. *Soit une paire d'entrées* (x,y) *et* (x',y') *de* π *ayant même transcription* t. *Alors* (x',y) *et* (x,y') *ont aussi même transcription, de plus cette transcription est* t.

En particulier, les sorties de π *sur les entrées* (x,y), (x',y), (x,y') *et* (x',y') *sont identiques.*

Preuve (esquisse). Considérons deux paires d'entrées (x,y) et (x',y') ayant même transcription $t = m(x,y) = m(x',y')$. Posons $t = t_1 t_2 \ldots$ avec $t_i = m_i(x,y)$. Par récurrence sur i, on montre que le i-ème message de π sur (x',y) est t_i, et de façon similaire sur (x,y').

Puisque la sortie $out(x,y)$ ne dépend que de la transcription et x, ou de la transcription et y, on en déduit que $out(x,y) = out(x',y)$, puis que $out(x',y) = out(x',y')$, soit au final que $out(x,y) = out(x',y')$. □

Cette propriété simple permet de montrer des bornes inférieures importantes, comme le théorème suivant.

Théorème 3.2. *Soient X, Y deux ensembles finis. Soient f une fonction définie sur $X \times Y$ et $S \subseteq X \times Y$ tels que*

1. *f est constante sur S ;*
2. *tout $(x,y),(x',y') \in S$ satisfait $f(x,y) \neq f(x',y)$ ou $f(x,y) \neq f(x,y')$.*

Alors

$$\mathsf{D}(f) \geq \lceil \log(1+|S|) \rceil.$$

Démonstration. Soit $z = f(x,y)$, pour un élément $(x,y) \in S$ quelconque, la valeur que f prend sur S. La contraposée du Lemme 3.1 et l'hypothèse (2) impliquent que chaque paire d'éléments $(x,y),(x',y') \in S$ a nécessairement une transcription différente. Donc il faut au moins $|S|$ transcriptions différentes pour avoir un protocole qui calcule correctement f.

Puisque la fonction f n'est pas constante, le Lemme 3.1 implique aussi qu'il faut au moins une transcription de plus pour calculer f, d'où le résultat. □

Le théorème 3.2 permet de montrer que

Lemme 3.3.

$$\mathsf{D}(\textsc{Equality}_n) = \mathsf{D}(\textsc{Disjointness}_n) = n+1.$$

Démonstration. La majoration est obtenue par le protocole trivial du Lemme **??**. Pour obtenir la minoration il suffit de considérer pour $\textsc{Equality}_n$ l'ensemble $S = \{(x,x) : x \in \{0,1\}^n\}$, et pour $\textsc{Disjointness}_n$ l'ensemble $S = \{(x,\overline{x}) : x \in \{0,1\}^n\}$, où $\overline{x}$ est obtenu en prenant la négation de tous les bits de x. Dans les deus cas, l'ensemble S est de taille 2^n et satisfait les hypothèses du théorème 3.2. Par conséquent, la quantité $\lceil \log(1+2^n) \rceil = n+1$ est un minorant de chacune des complexités. □

Cet autre lemme permet d'établir une borne inférieure sur $\mathsf{D}(f)$ à partir de propriétés algébriques de M_f. Notons $\delta_z(f)$ la fonction qui prend la valeur 1 en (x,y) si $z = f(x,y)$ et la valeur 0 sinon.

Théorème 3.4. *Soient X, Y deux ensembles finis, et $f : X \times Y \mapsto Z$.*

$$\mathsf{D}(f) \geq \left\lceil \log \left(\sum_{z \in Z} \operatorname{rang}(M_{\delta_z(f)}) \right) \right\rceil.$$

Démonstration. Fixons un protocole π pour f. Etant donnée une transcription t possible de π, appelons $R_t = \{(x,y) \in X \times Y : m(x,y) = t\}$. Nous interprèterons aussi R_t comme la matrice booléenne indexée par $x \in X$ et $y \in Y$, et prenant la valeur 1 en $(x,y) \in R_t$ et 0 sinon. D'après le lemme 3.1, nous savons que R_t est un rectangle, et que toute entrée de R_t mène vers une même sortie $out(t)$. Pour $z \in Z$, définissons $S_z := \{t : out(t) = z\}$. Alors

$$M_{\delta_z(f)} = \sum_{t \in S_z} R_t.$$

Le théorème du rang nous dit alors que

$$\operatorname{rang}(M_{\delta_z(f)}) \leq \sum_{t \in S_z} \operatorname{rang}(R_t) = |S_z|,$$

où l'égalité provient du fait que chaque R_t est non nul, et donc de rang au moins 1, pour $t \in S_z$.

La preuve se conclut en observant que $\sum_{z \in Z} |S_z|$ représente le nombre de transcriptions différentes du π peut transcrire lorsque son entrée (x, y) parcours $X \times Y$. Par conséquent sa complexité est au moins

$$\mathsf{C}(\pi) \geq \left\lceil \log \left(\sum_{z \in Z} |S_z| \right) \right\rceil .$$

□

Pour le problème $\textsc{Equality}_n$, le théorème 3.4 nous permet de redémontrer que

$$\mathsf{D}(\textsc{Equality}_n) \geq n + 1.$$

En effet, observons que $M_{\delta_1(f)}$ est la matrice identité de taille $2^n \times 2^n$, et est donc de rang n. De plus $M_{\delta_0(f)}$ n'est pas nulle, et donc $\mathsf{D}(\textsc{Equality}_n) > n$, ou encore $\mathsf{D}(\textsc{Equality}_n) \geq n+1$ puisque $\mathsf{D}(\textsc{Equality}_n)$ est une valeur entière.

Exemple 3.5. *La valeur médiane d'une famille $z_1, z_2, \ldots$ d'éléments de $[n]$ est définie par*

$$\text{médiane}(z) = \min\{v \in [n] : |\{i : z_i \geq v\}| \leq |\{i : z_i \leq v\}|\}.$$

Pour $X = Y = [n]^n$, le protocole ci dessous permet de calculer par une recherche dichotomique la valeur médiane de $(x, y) \in X \times Y$ avec une complexité de communication de $\lceil \log n \rceil (\lceil \log n \rceil + 1)$.

— *Partir de $a = 1$, $b = n$*
— *Pour i allant de 1 à $\lceil \log n \rceil$*
Alice et Bob :
— *Calculer $v = \lceil (a + b)/2 \rceil$*
Alice :
— *Calculer et envoyer $m_{2i-1} := |\{i : x_i \geq v\}|$*
Bob :
— *Calculer $k := m_{2i-1} + |\{i : x_i \geq v\}|$*
— *Si $k \leq n$ alors envoyer $m_{2i} := 0$*
Sinon envoyer $m_{2i} := 1$
Alice et Bob :
— *Si $m_{2i} = 0$ alors affecter $b := v$*
— *Sinon affecter $a := v$*
Fin du protocole La sortie du protocole est out $:= v$

4 Protocoles probabilistes

On procède de façon analogue pour les communications unilatérales probabilistes en définissant $\overrightarrow{\mathsf{R}}_{\varepsilon}^{\mathsf{pub}}(f)$ et $\overrightarrow{\mathsf{R}}_{\varepsilon}(f)$.

Obtenir des minorants de la complexité de communication probabiliste est beaucoup plus complexe. Nous verrons plus loin une méthode générique pour cela. En attendant voici quelques premiers résultats obtenus relativement simplement. A l'aide d'arguments plus sophistiqués, il est possible de supprimer le facteur logarithmique du Lemme 4.1.

Lemme 4.1.

$$\overrightarrow{\mathsf{R}}_{\frac{1}{3n}}(\textsc{Index}_n) \geq n, \quad \overrightarrow{\mathsf{R}}_{\frac{1}{3}}(\textsc{Index}_n) \geq n/\log n.$$

Démonstration. La deuxième minoration se déduit de la première comme suit. Etant donné un protocole π pour $\textsc{Index}_n$ avec erreur $1/3$, il suffit d'exécuter $\Theta(\log n)$ fois π, mais avec des choix aléatoires différents, pour obtenir une erreur au plus $1/(3n)$. En effectuant ces répétitions en parallèle le protocole reste uniliatéral.

Pour montrer la première minoration, nous allons montrer qu'il est possible de dérandomiser tout protocole π pour $\textsc{Index}_n$ d'erreur $1/(3n)$, tout en conservant sa complexité. Le lemme 2.3 permet alors de conclure. Soient r_A et r_B les bits aléatoires respectifs d'Alice et Bob. Le protocole déterministe est décrit par le protocole suivant :

- `Alice choisit` r_A `tel que` $\Pr_{r_A,r_B}[\exists j \in [n] : P(m(x,r_A),j,r_B) \neq x_j]$ `est minimal.`
- `Alice envoie` $t := m(x,r_A)$ `à Bob.`
- `Bob calcule la valeur majoritaire de` $out(t,i,r_B)$`, lorsque` r_B `parcourt tous les choix aléatoires possibles.`

Par hypothèse, pour tout $x \in \{0,1\}^n$ et $j \in [n]$,

$$\Pr_{r_A,r_B}[out(m(x,r_A),j,r_B) \neq x_j] \leq \frac{1}{3n}.$$

Par sommation (union-bound), on obtient pour tout $x \in \{0,1\}^n$,

$$\Pr_{r_A,r_B}[\exists j \in [n] : out(m(x,r_A),j,r_B) \neq x_j] \leq \frac{1}{3}.$$

Donc l'aléa r_a choisi par Alice et le message $t = m(x,r_A)$ envoyé à Bob satisfont nécessairement

$$\Pr_{r_B}[\exists j \in [n] : out(t,j,r_B) \neq x_j] \leq \frac{1}{3}.$$

Le message t permet de conclure pour toute entrée $i \in [n]$ de Bob. En effet, pour la valeur particulier $j = i$, l'inégalité précédente entraîne

$$\Pr_{r_B}[out(t,i,r_B) \neq x_i] \leq \frac{1}{3}.$$

Ainsi x_i est bien la valeur majoritaire de $out(t,i,r_B)$ lorsque r_B parcourt tous les choix aléatoires possibles de Bob. A noter que cette étape, ainsi que celle du choix de r_A, sont potentiellement longues mais déterministes et ne nécessitent aucune communication. □

5 Applications

Nous allons réduire différents problèmes déjà vus précédemment à des problèmes de complexité de communication. Nous pourrons en tirer des informations très utiles, notamment des résultats d'optimalité. Rappelons ici les résultats mentionnés précédemment :

.	$\overrightarrow{\mathsf{D}}(f)$	$\mathsf{D}(f)$	$\overrightarrow{\mathsf{R}}_{1/3}(f)$
$\textsc{Index}_n$	n	$\log(n)+1$	$\Theta(n)$
$\textsc{Equality}_n$	n	$n+1$	$O(\log n)$
$\textsc{Disjointness}_n$	n	$n+1$	

Considérons le problème de déterminer dans un flux de n entiers $a_1, a_2, \ldots$ de $[m]$, la plus grande multiplicité de ces entiers, notée aussi F^∞. Nous allons plus tard relier le calcul de F^∞ en streaming à celui du calcul d'une autre fonction (INDEX) en complexité de communication.

D'abord la complexité en mémoire d'un algorithme de streaming est simplement reliée à la complexité de communication comme suit.

Lemme 5.1. *Soit f une fonction définie sur $X \times Y$ avec $X = Y = \{0,1\}^n$. Alors tout algorithme de streaming à une passe doit utiliser une mémoire $s(n) \geq \overrightarrow{\mathsf{D}}(f)$ s'il est déterministe, et $s(n) \geq \overrightarrow{\mathsf{R}}_\varepsilon(f)$ s'il est probabiliste et d'erreur ε.*

De plus, tout algorithme de streaming à $p(n)$ passes doit utiliser une mémoire $s(n) \geq \mathsf{D}(f)/(2p(n))$ s'il est déterministe, et $s(n) \geq \mathsf{R}_\varepsilon(f)/(2p(n))$ s'il est probabiliste et d'erreur ε.

Ce lemme permet de montrer plusieurs compris mémoire et nombre de passes en streaming.

Théorème 5.2. *Tout algorithme de streaming déterministe à $p(n)$ passe, de mémoire $s(n)$, qui permet de décider si deux chaînes consécutives de n bits sont égales nécessite une mémoire $s(n) \geq n/(2p(n))$.*

Cependant, comme nous allons le voir, il est parfois nécessaire d'effectuer une réduction plus complexe, afin de transformer la fonction à calculer par un algorithme de streaming en une autre fonction plus adaptée à la complexité de communication.

Lemme 5.3. *Soit A un algorithme de streaming déterministe à une passe et de mémoire $s(n)$ permettant de calculer une valeur v telle que $|v - \mathrm{F}^\infty| < \mathrm{F}^\infty/3$ sur un flux de $n+1$ de $[2n]$. Alors $\overrightarrow{\mathsf{D}}(\textsc{Index}_n) \leq s(n)$.*

De plus, si A est probabiliste avec erreur ε, alors $\overrightarrow{\mathsf{R}}_\varepsilon(\textsc{Index}_n) \leq s(n)$.

Démonstration. En utilisant l'algorithme de streaming A, nous allons construire un protocole π unilatéral pour $\textsc{Index}_n$, où Alice reçoit l'entrée $x \in \{0,1\}^n$, et Bob l'entrée $i \in [n]$. La réduction est déterministe :

- Alice simule l'algorithme A sur le début du stream formé des entiers j tels que $x_j = 1$.
- Alice continue de simuler l'algorithme A sur des entiers $n < i' \leq 2n$ distincts jusqu'à avoir traité exactement n valeurs.
- Alice envoie la mémoire de l'algorithme A à Bob.
- Bob termine la simulation de l'algorithme A sur la fin du stream formée de l'unique entier i, et calcule la valeur v renvoyée par A.
- Si $v \geq 1/3$, alors Bob renvoie $out := 1$; sinon Bob renvoie $out := 0$

Pour prouver que le protocole a la même erreur que l'algorithme A (c'est-à-dire 0 si A est déterministe, et ε sinon), il suffit de remarque que $F^\infty = 1$ lorsque $x_i = 0$, et que $F^\infty = 2$ sinon. Lorsque v satisfait la promesse $|v - \mathrm{F}^\infty| < \mathrm{F}^\infty/3$, on conclue en observant que $v < 1/3$ lorsque $x_i = 0$, et $v > 1/3$ sinon. □

Puisque $\overrightarrow{\mathsf{R}}_{1/3}(\textsc{Index}_n) = \Theta(n)$, nous en déduisons le théorème suivant.

Théorème 5.4. *Tout algorithme de streaming probabiliste à une passe, de mémoire $s(n)$ et erreur $\varepsilon \leq 1/3$ permettant de calculer une valeur v telle que $|v - \mathrm{F}^\infty| < \mathrm{F}^\infty/3$ sur un flux de $n+1$ entiers de $[2n]$ nécessite une mémoire $s(n) = \Theta(n)$.*

Afin de généraliser ce résultat aux passes multiples, une réduction à un autre problème de communication doit être considérée. En effet, $\mathsf{D}(\textsc{Index}_n) = O(\log n)$. Nous considérons donc maintenant une nouvelle réduction à la fonction $\textsc{Disjointness}_n$.

Lemme 5.5. *Soit A un algorithme de streaming déterministe à $p(n)$ passes et de mémoire $s(n)$ permettant de calculer une valeur v telle que $|v - \mathrm{F}^\infty| < \mathrm{F}^\infty/3$ sur un flux de $2n$ entiers de $[3n]$. Alors $\mathsf{D}(\textsc{Disjointness}_n) \leq 2p(n)s(n)$.*

De plus, si A est probabiliste avec erreur $\varepsilon \leq 1/3$, alors $\mathsf{R}_\varepsilon(\textsc{Disjointness}_n) \leq 2p(n)s(n)$.

Démonstration. La preuve est très similaire à celle du lemme 5.3. La réduction est toujours déterministe :

- Alice simule l'algorithme A sur le début du stream formé des entiers i tels que $x_i = 1$, et complété avec des entiers $n < i' \leq 2n$ distincts afin d'obtenir n valeurs.
- Bob simule l'algorithme A sur la fin du stream formé des entiers i tels que $y_i = 1$, et complété avec des entiers $2n < i' \leq 3n$ distincts afin d'obtenir n valeurs.
- Au total le stream comporte $2n$ entiers de $[3n]$, les joueurs simulant à tour de rôle l'algorithme durant ses $p(3n)$ passes.
- A la fin de la simulation, Alice et Bob calculent la valeur v renvoyée par A.
- Si $v \geq 1/3$, alors ils décident $out := 1$; sinon $out := 0$

Au total $2p(n)s(n)$ bits sont échangés. Pour prouver que le protocole a la même erreur que l'algorithme A, il suffit de remarque que $F^\infty \leq 1$ lorsque $\textsc{Disjointness}_n(x, y) = 0$, et que $F^\infty = 2$ sinon. Lorsque v satisfait la promesse $|v - \mathrm{F}^\infty| < \mathrm{F}^\infty/3$, on conclue en observant que $v < 1/3$ lorsque $\textsc{Disjointness}_n(x, y) = 0$, et $v > 1/3$ sinon. □

Nous obtenons ainsi le résultat suivant qu'il est possible de généraliser aux protocoles probabilistes.

Théorème 5.6. *Tout algorithme de streaming déterministe à $p(n)$ passes, de mémoire $s(n)$ permettant de calculer une valeur v telle que $|v - \mathrm{F}^\infty| < \mathrm{F}^\infty/3$ sur un flux de $2n$ entiers de $[3n]$ nécessite une mémoire $s(n) \geq (n+1)/(2p(n))$.*