Université de Montréal

Complexité de la communication sur un canal avec délai

par
Rébecca Lapointe

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures et postdoctorales
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique

Février, 2014

RÉSUMÉ

Nous introduisons un nouveau modèle de la communication à deux parties dans lequel nous nous intéressons au temps que prennent deux participants à effectuer une tâche à travers un canal avec délai d. Nous établissons quelques bornes supérieures et inférieures et comparons ce nouveau modèle aux modèles de communication classiques et quantiques étudiés dans la littérature. Nous montrons que la complexité de la communication d'une fonction sur un canal avec délai est bornée supérieurement par sa complexité de la communication modulo un facteur multiplicatif $d/\lg d$. Nous présentons ensuite quelques exemples de fonctions pour lesquelles une stratégie astucieuse se servant du temps mort confère un avantage sur une implémentation naïve d'un protocole de communication optimal en terme de complexité de la communication. Finalement, nous montrons qu'un canal avec délai permet de réaliser un échange de bit cryptographique, mais que, par lui-même, est insuffisant pour réaliser la primitive cryptographique de transfert équivoque.

Mots clés: complexité de la communication, relativité, bornes inférieures, complexité de la communication quantique, cryptographie, rondes en complexité de la communication, théorie de l'informaion.

ABSTRACT

Communication Complexity on a Delayed Channel

We introduce a new communication complexity model in which we want to determine how much time of communication is needed by two players in order to execute arbitrary tasks on a channel with delay d. We establish a few basic lower and upper bounds and compare this new model to existing models such as the classical and quantum two-party models of communication. We show that the standard communication complexity of a function, modulo a factor of $d/\lg d$, constitutes an upper bound to its communication complexity on a delayed channel. We introduce a few examples on which a clever strategy depending on the delay procures a significant advantage over the naïve implementation of an optimal communication protocol. We then show that a delayed channel can be used to implement a cryptographic bit swap, but is insufficient on its own to implement an oblivious transfer scheme.

Keywords: Relativity, Lower Bounds, Quantum Communication Complexity, Cryptography, Rounds in Communication Complexity, Information Theory.

TABLE DES MATIÈRES

LISTE DES FIGURES

NOTATION

$\lg r$	Logarithme en base 2 de r
$\|r\|$	Taille de la chaîne binaire r
$r[i]$	i^e bit de chaîne binaire r

À maman et papa.

REMERCIEMENTS

J'aimerais remercier mon directeur Alain Tapp, pour sa patience et sa disponibilité.

Merci à Louis Salvail, Gilles Brassard, ainsi qu'aux autres membres du Laboratoire d'informatique théorique et quantique (LITQ), pour leurs suggestions et corrections.

Je remercie mes parents, mes grands-parents, mes sœurs et mon neveu pour leurs encouragements soutenus.

Merci à Alex, Heinz, CSJ et PK pour le soutien psychologique.

Finalement, je remercie le Fond québécois de recherche nature et technologie (FQRNT) et Alain Tapp pour le soutien financier.

INTRODUCTION

La complexité de la communication examine la quantité d'information que deux participants doivent partager à l'aide d'un canal de communication afin de réaliser une certaine tâche. Ce modèle néglige le fait qu'un canal réaliste est lesté d'un délai. Nous introduisons un modèle différent dans lequel nous ne sommes plus intéressés à minimiser le nombre total de bits à communiquer, mais à minimiser le temps total de communication à travers un canal avec délai.

Le but de ce mémoire est avant tout de comparer un modèle de communication sur un canal avec délai avec les modèles de communication étudiés dans la littérature dans le monde classique puis quantique. Dans un deuxième temps nous étudions comment un délai peut avoir des utilités cryptographiques.

Mise en contexte

Les canaux de communications effectifs, par exemple la fibre optique, sont tous affublés d'un délai qui, même s'il est minuscule, n'est pas négligeable pour deux familles de problèmes ; les communication à distances et les communications à haute fréquence.

Si on passe une fibre optique d'indice de réfraction de 1,52 de Melbourne à Montréal, la traversée prend 0,08 secondes. Une connexion de 10Gb impliquerait que pendant ce temps, 80 Mb de données aurait été envoyés avant que le premier bit ne soit arrivé à destination. Les dernières avancées dans le domaine de la fibre optique produisent une connexion de jusqu'à 400 Gb par seconde [5]. À chaque augmentation de la capacité des canaux de communication, le délai de communication devient plus critique.

Pour réduire le délai, la seule possibilité est d'accélérer la communication. Or, sa vitesse a une limite théorique ; tel que prédit par le principe de la relativité restreinte developpé par Einstein dans [14], aucune communication ne peut se faire plus rapidement que la vitesse de la lumière. Ainsi, même dans un monde idéal, où l'information utiliserait le chemin le plus court à sa vitesse maximale de Montréal à Melbourne, par

exemple a travers un tube à vide à travers la Terre, un bit prendrait quand même 0,037 pour effectuer sa traversée.

Une conséquence concrète de ce délai apparait dans le système de guidage du rover Curiosity qui doit effectuer beaucoup de ses calculs localement, ou à l'aide des orbiteurs martiens, afin d'éviter certains obstacles plutôt que de se fier aux communications de téléguidage terrestres [32].

Même à de courtes distances, le délai pèse. De manière générale, limiter le délai de communication est crucial dans les systèmes en temps réel nécessitant beaucoup de rétroaction. On peut imaginer une expérience scientifique utilisant plusieurs instruments et un superordinateur dans laquelle le système doit réagir en fonctions de ses mesures. On suppose qu'un instrument est éloigné de deux mètres du superordinateur et que celui-ci a une puissance d'un pétaFLOPS. L'ordinateur a le temps d'effectuer 10 millions de calculs sur des nombres à virgules flottantes entre le moment de la prise de mesure et le moment où il reçoit sa valeur. Ce nombre ne descend qu'à cent mille calculs si on rapproche l'ordinateur à deux centimètres de l'instrument.

Le délai importe également dans les transactions financières à haute fréquence, pour lesquelles des pistes de solutions sont données par Alexander Wissner-Gross et Cameron Freer dans [40].

Nos contributions

Le but de ce mémoire est de définir formellement la complexité de la communication dans un modèle où le canal de communication comporte un délai.

Nous comparons ce modèle avec les modèles de communication à deux parties classique et quantique largement étudiés dans la littérature et en dérivons des bornes inférieures et supérieures sur le temps minimal de communication. Nous montrons que la complexité de la communication d'une fonction sur un canal standard borne supérieurement sa complexité sur un canal avec délai d modulo un facteur multiplicatif dans l'ordre

de $d/\lg d$.

Nous survolons ensuite la potentialité qu'un tel modèle offre dans un contexte cryptographique. Nous montrons comment simuler un échange de bit sur un canal avec délai et nous montrons qu'une garantie de délai en elle seule est insuffisante pour simuler un transfert équivoque.

Plan

Dans le chapitre 1, nous présentons les modèles de communication à deux parties classique et quantique. Nous amenons également le principe de messages, de protocoles probabilistes et de somme directe.

La définition formelle du modèle de complexité de la communication sur un canal avec délai se trouve au chapitre 2 et est accompagnée de bornes inférieures et supérieures de base. Nous montrons aussi comment implémenter un protocole sur un canal standard en un protocole sur un canal avec délai.

Dans le chapitre 3, nous montrons comment sauver un facteur logarithmique sur l'implémentation naïve d'un protocole de communication dans le modèle de communication avec délai.

Dans le chapitre 4 nous illustrons dans différents scénarios comment des protocoles dépendant du délai peuvent être plus efficaces que l'implémentation montrée dans le chapitre 3.

Nous étudions les considérations cryptographiques dans le chapitre 5. Nous montrons comment le délai peut être utilisé pour implémenter un échange de bit, mais aussi comment il est insuffisant, par lui-même, pour implémenter un transfert équivoque.

CHAPITRE 1

PRÉLIMINAIRES

Dans ce chapitre, nous introduisons divers concepts théoriques qui seront utilisés au cours de ce mémoire. Nous commençons par introduire le modèle de communication à deux parties et les bornes élémentaires sur la complexité de la communication.

On présente ensuite les protocoles probabilistes, le principe de rondes dans la communication, les problèmes de type somme directe et les protocoles de communication quantiques. On présente aussi certains résultats fondamentaux de chacun ainsi que des références vers des revues de littérature plus exhaustives.

1.1 Modèle de communication à deux parties

Énonçons d'abord le modèle standard de la communication à deux parties tel qu'introduit par Harold Abelson dans [2] et formalisé par Andrew Yao dans [41]. Dans ce modèle, deux participants Alice et Bob sont confrontés au problème suivant. Soit X, Y et Z des ensembles quelconques finis et soit une fonction arbitraire $f : X \times Y \longrightarrow Z$. Les deux participants souhaitent connaître l'évaluation de $f(x,y)$ sur deux entrées $x \in X$ et $y \in Y$ avec la restriction qu'Alice ne connaît que x et Bob ne connaît que y. Afin d'effectuer leur tâche, ils ont accès à un canal de communication binaire sans bruit.

Dans ce modèle, nous sommes intéressés à connaître la quantité d'information qui doit être transmise à travers le canal afin qu'Alice et Bob puissent évaluer la sortie de la fonction. Nous négligeons les calculs faits localement par chacune des parties. On fait l'hypothèse que ceux-ci ont un pouvoir calculatoire local illimité.

Bien que la complexité de la communication peut être étudiée à partir de fonctions définies de façon générale pour tout ensemble fini, nous nous restreignons à l'étude de fonctions booléennes. Nous forçons aussi l'entrée d'Alice à être une chaîne de bits de longueur n et l'entrée de Bob à être une chaîne de bits de longueur m. Formellement,

$f : \{0,1\}^n \times \{0,1\}^m \longrightarrow \{0,1\}$. Bien sûr, tout ensemble fini X et Y peut être encodé sous forme d'ensembles de chaînes binaires et on peut combiner le calcul de plusieurs fonctions binaires afin de calculer n'importe quelle fonction avec une image finie.

Ce modèle de communication à deux parties peut être généralisé à un nombre arbitraire de participants, mais nous nous limiterons à deux participants dans ce mémoire.

1.1.1 Applications du modèle

Aussi simple qu'il paraît, le modèle de communication à deux parties a des utilités dans domaines variés comme, entre autres, l'intégration à très grande échelle (VLSI) ([39], [31]), l'étude des arbres de décisions ([34]) et les circuits booléens ([24]).

Pour un état de l'art du modèle de communication à deux parties et ses applications, nous référons le lecteur au livre d'Eyal Kushilevtitz et Noam Nisan ([30]) et à la revue de littérature de Christos Papadimitriou et Michael Sipser ([35]).

1.1.2 Définitions

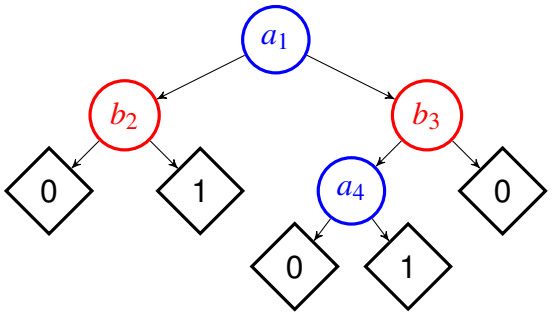
Définition 1. [30] Un *protocole de communication* $\mathscr{P}$ est un arbre binaire dans lequel chaque nœud interne v est étiqueté par une fonction $a_v : \{0,1\}^n \to \{0,1\}$ ou par une fonction $b_v : \{0,1\}^m \to \{0,1\}$ et où chaque feuille est étiquetée par un bit.

La sortie du protocole $\mathscr{P}$ sur entrée (x,y) est donnée par l'étiquette de la feuille qui est atteinte lorsqu'on suit le chemin à partir de la racine et en se déplaçant à gauche lorsque $a_v(x) = 0$ ou $b_v(y) = 0$ et à droite lorsque $a_v(x) = 1$ ou $b_v(y) = 1$. Elle est notée par $\mathscr{P}(x,y)$. Les nœuds internes étiquetés par une fonction a_v représentent un bit envoyé par Alice et ceux étiquetés par une fonction b_v représentent un bit envoyé par Bob. Ainsi, lors de l'exécution du protocole, les participants n'ont qu'à descendre dans l'arbre à chaque bit communiqué afin de déterminer qui envoie le prochain bit et par quelle fonction il est donné. Lorsqu'ils atteignent une feuille, ils apprennent la sortie du protocole.

On dit qu'un *protocole $\mathscr{P}$ calcule une fonction f* si pour toute paire d'entrées (x,y), la sortie de $\mathscr{P}$ est $f(x,y)$.

Dans ce mémoire, on se limitera à dire protocole plutôt que protocole de communication.

Exemple 1 (Exemple de protocole [30])**.** Dans la figure 1.1, Les nœuds en bleu sont étiquetés par une fonctions associée à Alice, ceux en rouge à Bob et les feuilles sont la sortie du protocole. Le domaine $X \times Y$ est $\{x,x',x'',x'''\} \times \{y,y',y'',y'''\}$ et les fonctions de chaque nœuds sont données dans le tableau de droite. Ainsi, sur entrée (x'',y), Alice envoie le premier bit $a_1(x'') = 1$, indiquant un déplacement vers le fils droit. Bob répond $b_3(y) = 1$. Le protocole ayant atteint une feuille, les deux participants connaissent la sortie protocole, soit $f(x'',y) = 0$.



$a_1(x) = 0$	$b_2(y) = 0$
$a_1(x') = 0$	$b_2(y') = 0$
$a_1(x'') = 1$	$b_2(y'') = 0$
$a_1(x''') = 1$	$b_2(y''') = 1$
$b_3(y) = 1$	$a_4(x) = 0$
$b_3(y') = 0$	$a_4(x') = 0$
$b_3(y'') = 0$	$a_4(x'') = 0$
$b_3(y''') = 0$	$a_4(x''') = 1$

Figure 1.1 – Arbre représentant un protocole

Définition 2. Le *coût d'un protocole $\mathscr{P}$ sur entrée* (x,y) est le nombre de bits échangés par Alice et Bob et est donné par la longueur du chemin parcouru sur entrée (x,y) et le *coût du protocole $\mathscr{P}$* qui calcule la fonction f est le nombre de bit maximum échangés par Alice et Bob sur toutes les entrées possibles. Il est donné par la hauteur de l'arbre et est noté $C_{\mathscr{P}}(f)$.

Exemple 2. Le plus long chemin de la racine à une feuille dans le protocole de l'exemple 1 est atteint par l'entrée (x'', y'), lui donnant un coût de 3.

Exemple 3 (MOY)**.** [30] Alice et Bob ont comme entrée respectivement X et Y, deux sous-ensembles de $\{1, 2, \ldots, n\}$ et souhaitent calculer $\texttt{MOY}(X, Y)$, la moyenne du multiensemble $x \uplus y$. Dans cet exemple, la fonction n'est pas booléenne.

Un protocole répondant à cette tâche est le protocole dans lequel Alice envoie la somme des éléments de X ainsi que sa taille à Bob. Bob envoie ensuite la somme des éléments de Y ainsi que sa taille à Alice. Chacun de leur côté Alice et Bob calculent la moyenne du multiensemble $x \uplus y$ donnée par $\frac{\sum X + \sum Y}{|X| + |Y|}$. Dans le pire cas, les sous-ensembles sont maximaux, $|X| = |Y| = n$, ce qui confère un coût de $2\left\lceil \lg \frac{n(n+1)}{2} \right\rceil + 2\lceil \lg n \rceil$ au protocole.

Définition 3. [30] La *complexité de la communication* d'une fonction f, notée $C(f)$ est la complexité de la communication que prend le protocole calculant f le plus efficace, i.e.

$$C(f) = \min_{\mathscr{P}} \{C_{\mathscr{P}}(f) | \mathscr{P} \text{ est un protocole qui calcule } f\}.$$

Définition 4. Réciproquement, on dira qu'un protocole $\mathscr{P}$ qui calcule f est *optimal* si sa complexité est dans l'ordre de la complexité de communication de f.

$$C_{\mathscr{P}}(f) \in \mathscr{O}\left(C(f)\right).$$

On dira qu'un protocole est *exactement optimal* si sa complexité est égale à la complexité de communication de f.

$$C_{\mathscr{P}}(f) = C(f).$$

1.1.3 Bornes élémentaires

Dans [41], Andrew Yao montre que la complexité de la communication d'une fonction est bornée supérieurement par

Lemme 1.

$$C(f) \leq n+1$$

Démonstration. Le protocole 1 calcule f. Alice transmet n bits puis Bob répond avec un seul bit.

Protocole 1 : Protocole universel calculant la fonction f sur entrée (x,y).

1 Alice envoie son entrée x à Bob;

2 Bob calcule $s = f(x,y)$ et envoie s à Alice;

3 Alice et Bob retournent s;

□

Les bornes inférieures sont plus difficiles à déterminer. On connaît tout de même quelques bornes inférieures, dont une utilisant les matrices de fonctions.

Définition 5. La *matrice M associée à la fonction* f est une matrice 2^n par 2^m où $M_{i,j} = f(i,j)$ pour toute paire $(i,j) \in \{0,1\}^n \times \{0,1\}^m$

Exemple 4 (Exemple de matrice de fonction)**.** La matrice 1.2 est la matrice de la fonction calculée par le protocole illustré dans la figure 1.1

	y	y'	y''	y'''
x	0	0	0	1
x'	0	0	0	1
x''	0	0	0	0
x'''	0	1	1	1

Figure 1.2 – Matrice la fonction calculée par le protocole 1.1

Lemme 2. *[41]*

$$C(f) \geq \lg rang(f)$$

Où rang(f) est le rang de la matrice de la fonction f.

1.2 Complexité de la communication de protocoles probabilistes

Dès sa formalisation par Andrew Yao dans [41], le modèle de communication est étendu à un modèle où Alice et Bob peuvent accéder à de l'aléa. Nous présentons différentes définitions de la complexité de la communication probabiliste provenant du livre de Eyal Kushilevitz et Noam Nisan [30].

1.2.1 Définitions

Soit Alice et Bob deux participants ayant accès à une chaîne aléatoire de longueur arbitraire, respectivement r_A et r_B, choisie indépendemment dans une distribution de probabilité donnée. Ils ont aussi, comme précédemment, leur entrée respective $x \in \{0,1\}^n$ et $y \in \{0,1\}^m$.

Définition 6. Un *protocole probabiliste* $\mathscr{P}$ est défini par un arbre comme celui de la définition 1, avec l'exception que les fonctions d'Alice dépendent de x et de r_A et que les fonctions de Bob dépendent de y et de r_B. Sa sortie, sur entrée (x,y) est la feuille atteinte par le quadruplet (x,r_A,y,r_B) et est notée $\mathscr{P}(x,y)$.

On remarque que la sortie du protocole change en fonction des chaînes aléatoires. Soit f une fonction binaire sur $X \times Y$ deux ensembles finis et $0 < \varepsilon < 1$

Définition 7. On dit que $\mathscr{P}$ *calcule* f *sans erreur* si pour toute entrée $(x,y) \in X \times Y$, on a

$$\Pr[\mathscr{P}(x,y) = f(x,y)] = 1.$$

Définition 8. On dit que $\mathscr{P}$ *calcule* f *avec erreur* ε si pour toute entrée $(x,y) \in X \times Y$, on a

$$\Pr[\mathscr{P}(x,y) = f(x,y)] \geq 1 - \varepsilon.$$

Définition 9. On dit que $\mathscr{P}$ *calcule* f *avec erreur* ε *d'un côté* si pour toute entrée $(x,y) \in$

$X \times Y$ telle que $f(x,y) = 0$, on a

$$\Pr[\mathscr{P}(x,y) = f(x,y)] = 1$$

et pour toue entrée (x,y) telle que $f(x,y) = 1$, on a

$$\Pr[\mathscr{P}(x,y) = f(x,y)] \geq 1 - \varepsilon.$$

Définition 10. Le *temps de calcul dans le pire cas de $\mathscr{P}$ sur entrée* (x,y) est le nombre maximum de bits échangé entre Alice et Bob lors de l'exécution de $\mathscr{P}$ sur (x,y) pour tout choix de chaînes aléatoires r_A et r_B. Le *coût du pire cas de $\mathscr{P}$* est le maximum, sur toute entrée $(x,y) \in X \times Y$, du temps de calcul dans le pire cas de $\mathscr{P}$ sur (x,y).

Définition 11. Le *temps de calcul moyen d'un protocole probabiliste $\mathscr{P}$ sur entrée* (x,y) est le nombre espéré de bits échangés entre Alice et Bob lors de l'exécution de $\mathscr{P}$ sur entrée (x,y) pour tout choix de chaînes aléatoires r_A et r_B. Le *coût moyen de $\mathscr{P}$* est le maximum, sur toute entrée $(x,y) \in X \times Y$, du temps de calcul moyen de $\mathscr{P}$ sur (x,y).

Pour chacune de ses définitions, on aura les mesures de complexité suivantes :

Définition 12. $R_0(f)$ est le coût moyen minimal d'un protocole probabiliste qui calcule f sans erreur.

Définition 13. $R_\varepsilon(f)$ est le pire coût minimal d'un protocole probabiliste qui calcule f avec erreur ε. Nous utilisons la convention que $R(f) = R_{1/3}(f)$.

Définition 14. $R^1_\varepsilon(f)$ est le pire coût minimal d'un protocole probabiliste qui calcule f avec erreur ε d'un côté.

1.3 Messages

En plus de s'intéresser à la quantité d'information à transmettre pour calculer une fonction, on peut s'intéresser au nombre de messages échangés entre Alice et Bob. Un

message est un mot constitué de tous les bits envoyés par un participant sans intervention du deuxième. Le concept de messages nous permet de représenter un protocole sous forme d'un diagramme de communication, comme fait dans le protocole 1, constitué de deux messages.

1.3.1 Définitions

Définition 15. [35] Un *protocole de k messages* est un protocole dans lequel, sur toutes les entrées, il y a un maximum de k messages échangés entre les participants.

On note que pour toute fonction, il est toujours possible de construire un protocole de deux messages, soit un protocole dans lequel Alice envoie toute son entrée, et Bob retourne l'évaluation de la fonction.

Dans ce mémoire, nous aurons besoin de définitions spécifiques.

Définition 16. Le *coût en messages* d'un protocole $\mathscr{P}$ qui calcule f, est le nombre maximum de messages entre Alice et Bob sur toutes les entrées possibles.

Définition 17. Le *nombre de messages minimum d'un protocole optimal pour f* est le coût en messages minimal sur tout les protocoles optimaux qui calculent f. Nous le notons $\Gamma(f)$.

$$\Gamma(f) = \min_{\mathscr{P}} \{\Gamma_{\mathscr{P}}(f) | \mathscr{P} \text{ est optimal pour calculer } f\}$$

Il sera parfois utile de décortiquer un protocole par ses messages. Nous utilisons notation suivante : $\gamma_i(\mathscr{P}, x, y)$ est le i^e message envoyé lors de l'exécution du protocole $\mathscr{P}$ sur entrée (x, y).

1.3.2 Résultat fondammental

Noam Nisan et Avi Widgerson ont montré dans [34] qu'il peut y avoir une explosion exponentielle entre la complexité d'une fonction de k message et sa complexité si on la

restreint à $k-1$ messages.

1.4 Somme directe

1.4.1 Définitions

Définition 18. Le problème de *somme directe* dans le contexte de la complexité de la communication est présenté dans [23] de la façon suivante : Alice et Bob ont chacun ℓ entrées $x_1, x_2, \ldots, x_\ell \in \{0,1\}^n$ et $y_1, y_2, \ldots, y_\ell \in \{0,1\}^m$. Ils souhaitent calculer $f(x_1, y_1)$, $f(x_2, y_2)$,…,$f(x_\ell, y_\ell)$. On dénote la complexité de la communication du calcul de la fonction f sur ℓ paires d'entrées par $C(f^\ell)$.

Le calcul de ces fonctions peut être fait en calculant f successivement sur chacune des entrées. Une première borne est donc donnée par

$$C(f^\ell) \leq \ell \times C(f).$$

1.4.2 Résultats fondammentaux

Rien ne garantit que cette borne est optimale. En effet, Tomas Feder, Eyal Kushilevitz, Moni Naor et Noam Nisan donnent dans [16] des exemples de fonctions partielles et de fonctions probabilistes pour lesquelles la complexité de la communication de f^ℓ est plus facile que le calcul successif de f sur les ℓ entrées par un facteur logarithmique. Dans le cas de fonctions booléennes, la complexité de la communication de la somme directe est bornée inférieurement par l'équation suivante :

$$C(f^\ell) \in \Omega\left(\ell\left(\sqrt{C(f)} - \lg n - \mathcal{O}(1)\right)\right).$$

On peut être intéressé à calculer une fonction sur f^ℓ plutôt que de calculer les ℓ résultats. On peut par exemple, vouloir calculer $f(x_1, y_1) \wedge f(x_2, y_2)$. Pour ce type de problème, Dietzfelbinger et al. démontrent dans [12] qu'une borne inférieure pour f

donnée par le rang de sa matrice nous donne aussi une borne inférieure sur le $\wedge$ booléen :

$$\text{rang}(f \wedge g) = \text{rang}(f) \cdot \text{rang}(g).$$

1.5 Complexité de la communication de protocoles quantiques

Fort de l'apport qu'a le modèle de la communication à deux parties, il est pertinent de se demander comment la complexité change si on permet à Alice et Bob de communiquer des qubits ou encore, s'ils partagent une chaîne de qubits intriqués. Le qubit l'analogue quantique du bit. Plutôt que de représenter un état de base (soit 0, soit 1), le qubit est une superposition des états de base. Une particularité d'une chaîne de qubits est qu'elle peut être intriquée, permettant à deux participants qui la partagent d'effectuer des tâches de communication de manière plus efficace. Nous ne présentons pas les définitions et résultats fondamentaux du calcul quantique, référant le lecteur à des ouvrages d'introduction, tel le livre de Michael Nielsen et Isaac Chuang [33].

Pour un aperçu des résultats dans la complexité de la communication quantique, nous référons le lecteur à des revues de littérature sur le sujet ([6], [28]).

1.5.1 Définitions

Un protocole est dit quantique, selon le modèle de Cleve-Buhrman [10], si les participants ont accès à une chaîne de qubits intriqués et que les participants communiquent des bits classiques.

Soit $\mathscr{P}$ un protocole quantique, f une fonction binaire sur $X \times Y$ deux ensembles finis et $0 < \varepsilon < 1$. Tout comme dans le cas probabiliste, nous avons les définitions suivantes :

Définition 19. $Q_0(f)$ est le coût en bits classiques communiqués moyen minimal d'un protocole quantique qui calcule f sans erreur.

Définition 20. $Q_\varepsilon(f)$ est le pire coût minimal d'un protocole quantique qui calcule f avec erreur ε. Nous utilisons la convention que $Q(f) = Q_{1/3}(f)$.

Définition 21. $Q^1_\varepsilon(f)$ est le pire coût minimal d'un protocole quantique qui calcule f avec erreur ε d'un côté.

1.5.2 Résultats fondammentaux

Ran Raz a montré dans [37] qu'il peut y avoir une séparation exponentielle entre la complexité de la communication d'une fonction probabiliste avec promesse et sa complexité de la communication quantique.

Nous nous intéressons à un résultat moins puissant relatif au problème d'ensembles disjoints défini par

$$\begin{aligned} \texttt{ENSEMBLES\_DISJOINTS} : \{0,1\}^n \times \{0,1\}^n &\rightarrow \{0,1\} \\ (x,y) &\mapsto \bigvee_{i=1}^{n} (x[i] \wedge y[i]) \end{aligned}$$

Pour cette fonction, Bala Kalyanasundaram et Georg Schintger ont déterminé dans [22] que

$$R(\texttt{ENSEMBLES\_DISJOINTS}) \in \Theta(n).$$

Or, conjointement, les résultats de Razborov ([38]), Harry Buhrman, Richard Cleve et Avi Wigderson ([8]), Peter Høyer et Ronald de Wolf ([19]) et Scott Aaronson et Andris Ambainis ([1]) caractérisent exactement sa complexité de la communication quantique

$$Q(\texttt{ENSEMBLES\_DISJOINTS}) \in \Theta\left(\sqrt{n}\right),$$

conférant un avantage quadratique sur sa complexité de la communication classique dans le modèle pourvu d'aléa.

1.6 Sommaire

Le modèle de communication à deux parties étudie le nombre de bits échangés entre deux participants afin de répondre à une certaine tâche. Ce modèle peut être étendu dans sa forme déterministe ou probabiliste, classique ou quantique. Chacune de ces saveurs sera étendue dans un modèle où on s'intéresse plutôt au temps total nécessaire à ces deux participants pour répondre à la même tâche.

CHAPITRE 2

COMPLEXITÉ DE LA COMMUNICATION À DEUX PARTIES SUR UN CANAL AVEC DÉLAI

Nous introduisons un nouveau modèle qui est, à notre avis, plus réaliste dans un monde de communication à distance. Dans ce modèle, nos participants Alice et Bob sont séparés par une distance quelconque, mais ont accès à un canal de communication bidirectionnel de délai d sur lequel exactement un bit est transmis à chaque unité temporelle dans chacune des directions. Chaque bit prend d unités de temps pour atteindre sa destination. Par exemple, si Alice transmet un bit au temps t, Bob le recoit au temps $t+d$, soit après qu'Alice ait envoyé $d-1$ bits quelconques au canal. La donnée qui nous intéresse dans ce modèle n'est plus le nombre de bits échangés, mais le temps total d'exécution du protocole. Nous faisons l'hypothèse que les participants ont un pouvoir de calcul local illimité.

2.1 Définition

Un *protocole $\mathscr{P}$ avec délai d* sur entrée (x,y), où Alice connait $x \in \{0,1\}^n$ et Bob connait $y \in \{0,1\}^m$ est donné par les quatre ensembles de fonctions suivantes :

$$\begin{aligned}
\alpha_i &: \{0,1\}^n \times \{0,1\}^{\max\{0,i-d\}} \to \{0,1\},\\
A_i &: \{0,1\}^n \times \{0,1\}^{\max\{0,i-d\}} \to \{\perp,0,1\},\\
\beta_i &: \{0,1\}^m \times \{0,1\}^{\max\{0,i-d\}} \to \{0,1\} \text{ et}\\
B_i &: \{0,1\}^m \times \{0,1\}^{\max\{0,i-d\}} \to \{\perp,0,1\}.
\end{aligned}$$

Pour i un entier naturel, $\alpha_i(x,(y_1,\ldots,y_{i-d})) = x_i$, représente le bit envoyé par Alice sur le canal à l'instant i, il dépend de x et des bits envoyés par Bob au moins d unités de temps dans le passé. De la même manière, $\beta_i(y,(x_1,\ldots,x_{i-d})) = y_i$, représente le bit

envoyé par Bob sur le canal à l'instant i, il dépend de y et des bits envoyés par Alice au moins d unités de temps dans le passé.

La fonction $A_i(x,(y_1,\ldots,y_{i-d}))$ donne la sortie d'Alice à l'instant i. Tant qu'Alice n'a pas terminé le protocole, $A_i(\cdot,\cdot) = \perp$. Si Alice termine le protocole au temps i, $A_j(\cdot,\cdot)$ représente la sortie du protocole pour tout $j \geq i$. La fonction $B_j(y,(x_1,\ldots,x_{i-d}))$ est définie similairement. La sortie du protocole est toujours la même pour Alice et Bob.

Les fonctions A_i et B_i servent à illustrer qu'Alice et Bob ne connaissent pas la sortie du protocole au même moment. Nous dirons que le protocole se termine quand Alice et Bob ont terminé.

Dans le reste du mémoire, afin d'éviter la confusion entre les type de canaux de communication, un canal de communication sans délai sera appelé un canal de communication standard.

On peut visualiser un protocole sous la forme d'un diagramme de séquence, comme dans la figure 2.1. Pour éviter de surcharger le dessin, nous ne montrons pas tous les bits échangés. Dans l'exemple ci-dessous, Alice envoie deux bits à Bob, et sur réception, Bob envoie deux bits à Alice et termine. Alice termine quand elle reçoit ces derniers. Le coût du protocole sur entrée (x,y) est de $2d+1$.

Remarque. Nous utiliserons le terme protocole à la fois pour les protocoles de communication et pour les protocoles de communication sur un canal avec délai. Le contexte définira toujours de quel type de protocole il s'agit.

De nouveau, le modèle peut être généralisé à un nombre fini de participants. Nous nous limiterons à deux.

Définition 22. On dit qu'un protocole de communication sur un canal avec délai *calcule une fonction* f, si sur toute entrée (x,y), la sortie du protocole est $f(x,y)$.

Définition 23. Le *coût d'un protocole* $\mathscr{P}$ *sur un canal de délai* d d'une fonction f, notée $D_{\mathscr{P}}(f,d)$ est le temps maximum que prend l'exécution de $\mathscr{P}$ sur ses entrées. Il est donné par le plus petit indice i à partir duquel Alice et Bob ont terminé.

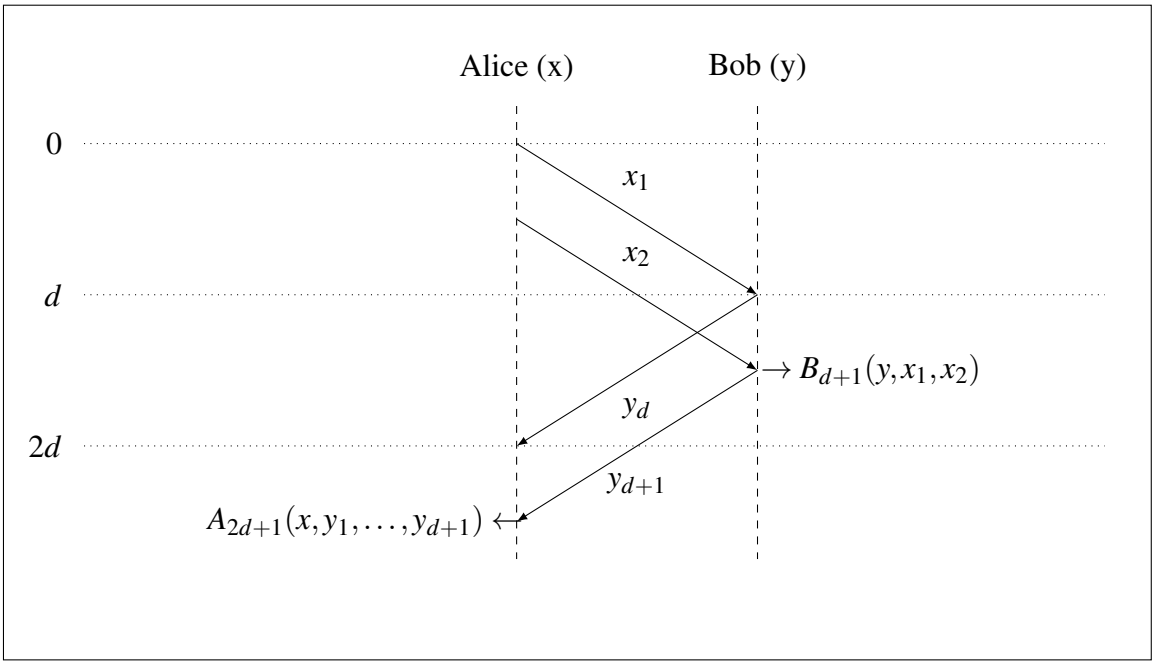


Figure 2.1 – Exemple de protocole sur un canal avec délai

Définition 24. La *complexité de la communication sur un canal de délai d* d'une fonction f, notée $D(f,d)$ est le temps que prend le protocole calculant f le plus efficacement, i.e.

$$D(f,d) = \min_{\mathscr{P}}\{D_{\mathscr{P}}(f,d) | \mathscr{P} \text{ est un protocole qui calcule } f\}.$$

Cet ensemble de définitions en poche, nous présentons une première borne sur la complexité de la communication sur un canal avec délai.

Théorème 1.

$$D(f,d) \leq \max\{m,n\} + d$$

Démonstration. Dans le protocole de la figure 2.2 la chaîne x met $n+d$ unités de temps pour arriver à Bob. Simultanément, la chaîne y met $m+d$ unités de temps pour arriver à Alice. □

Alternativement, on a

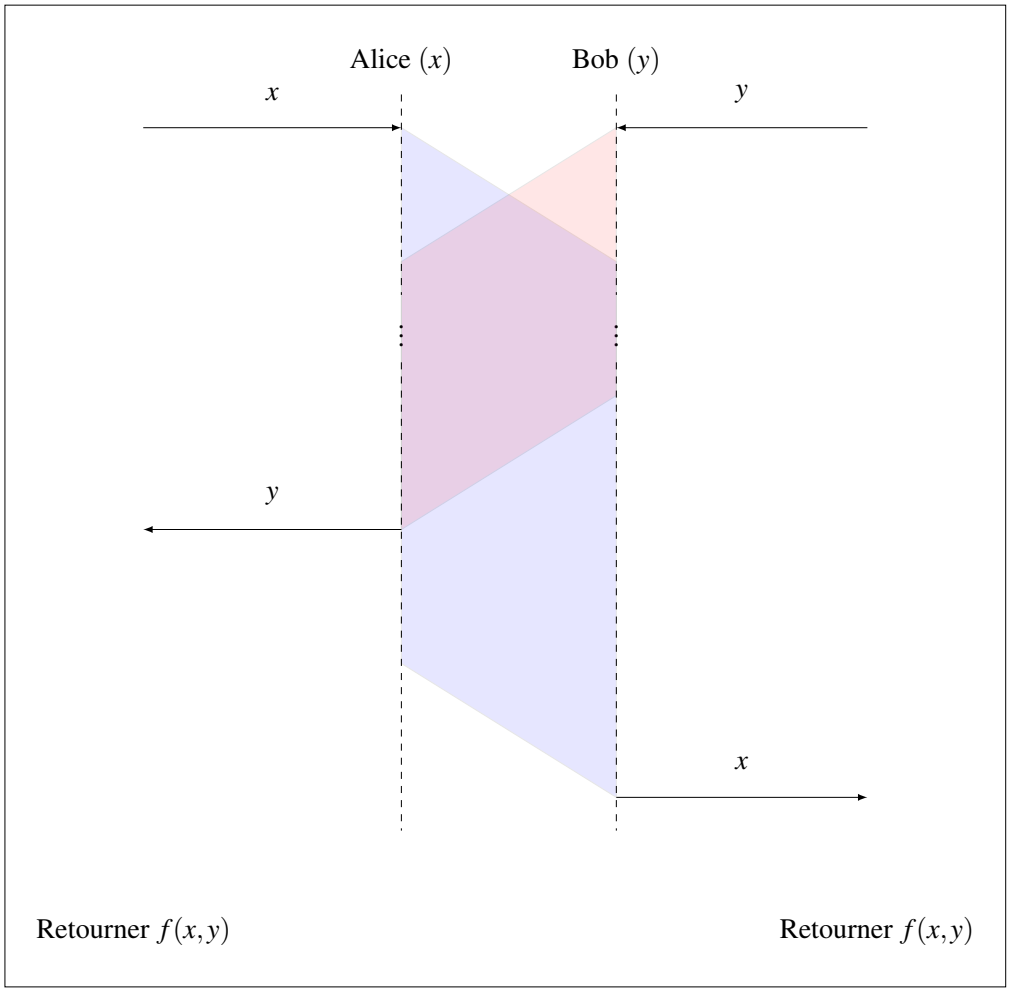


Figure 2.2 – Protocole universel sur un canal avec délai calculant une fonction f sur entrée (x,y)

Théorème 2.

$$D(f,d) \leq \min\{m,n\} + 2d$$

Démonstration. Si Alice a l'entrée la plus courte, le protocole suivant atteint cette complexité :

Protocole 2 : Protocole universel alternatif sur un canal avec délai calculant une fonction f sur entrée (x,y)

1 Alice envoie x à Bob;

2 Sur réception, Bob calcule $f(x,y)$ et retourne la réponse;

L'étape 1 prend $n+d$ unités de temps et l'étape deux ajoute d unités de temps. Le protocole est inversé si c'est Bob qui a l'entrée la plus courte. □

Remarque. Dans le cas où la taille des entrées diffère de plus de d bits, le protocole du théorème 2 offre une meilleure borne et vice-versa.

Remarque. La contrainte que chaque participant doit connaître le résultat de la fonction a des conséquences bien différentes dans le modèle de communication avec délai. Un participant qui communique $f(x,y)$ à l'autre n'ajoute qu'un bit à la complexité de la communication d'un protocole standard (comme dans la preuve du lemme 1). Dans le modèle avec délai, l'envoi de ce bit ajoute $d+1$ unités de temps à l'exécution du protocole.

2.2 Exemples de protocoles sur un canal avec délai

Montrons maintenant comment calculer certaines fonctions binaires qui reviennent souvent dans la littérature sur un canal de communication avec délai.

Définissons les fonctions `PRODUIT_INTERNE` et `ENSEMBLES_DISJOINTS`

$$\begin{aligned} \texttt{PRODUIT\_INTERNE} : \{0,1\}^n \times \{0,1\}^n &\rightarrow \{0,1\} \\ (x,y) &\mapsto \bigoplus_{i=1}^{n} (x[i] \wedge y[i]) \\ \texttt{ENSEMBLES\_DISJOINTS} : \{0,1\}^n \times \{0,1\}^n &\rightarrow \{0,1\} \\ (x,y) &\mapsto \bigvee_{i=1}^{n} (x[i] \wedge y[i]) \end{aligned}$$

Tout comme ÉGALITÉ, `PRODUIT_INTERNE` et `ENSEMBLES_DISJOINTS` sont des fonc-

tions de la forme $G_{i=1}^{n} f(x[i], y[i])$. Où $G : \{0,1\}^2 \to \{0,1\}$ est associative et $f : \{0,1\}^2 \to \{0,1\}$.

Où dans ÉGALITÉ, $G(z_1, z_2) = z_1 \wedge z_2$ et $f(a_k, b_k) = 1 \iff a_k = b_k$,

dans PRODUIT_INTERNE, $G(z_1, z_2) = z_1 \oplus z_2$ et $f(a_k, b_k) = a_k \wedge b_k$ et

dans ENSEMBLES_DISJOINTS, $G(z_1, z_2) = z_1 \vee z_2$ et $f(a_k, b_k) = a_k \wedge b_k$.

Nous définissons deux protocoles de communication sur un canal avec délai répondant à toutes les fonctions de cette forme.

Un premier protocole peut être bâti par l'envoi par chacune des parties de l'entièreté de leur entrée.

Protocole 3 : Protocole sur canal avec délai pour calculer $G_{i=1}^{n} f(x[i], y[i])$

1. Alice envoie x à Bob et Bob envoie y à Alice;
2. Ils calculent chacun de leur côté $G_{i=1}^{n} f(x[i], y[i])$ et en retournent la valeur;

Plutôt que d'envoyer leur entrée au complet, Alice et Bob pourraient chacun calculer la fonction G sur la moitié de leur entrée, puis partager le résultat.

Protocole 4 : Protocole sur canal avec délai pour calculer $G_{i=1}^{n} f(x[i], y[i])$

1. Alice envoie $x[1] \ldots x[\lfloor n/2 \rfloor]$ à Bob et Bob envoie $y[\lfloor n/2 \rfloor + 1] \ldots y[n]$ à Alice;
2. Sur réception, Alice calcule $a = G_{i=1}^{\lfloor n/2 \rfloor} f(x[i], y[i])$ et Bob calcule $b = G_{i=\lfloor n/2 \rfloor + 1}^{n} f(x[i], y[i])$;
3. Alice envoie a à Bob et Bob envoie b à Alice;
4. Alice et Bob retournent $G(x, y)$;

On observe que $D_{\text{protocole 3}}(f, d) = n + d$ et que $D_{\text{protocole 4}}(f, d) = \lceil n/2 \rceil + 2d$. Le protocole 4 est plus efficace que le protocole 3 dès que $n > 2d$, alors que la première stratégie est avantageuse lorsque $n < 2d$. Nous définissons un nouveau protocole qui utilise les avantages de chacun de ceux-ci.

Protocole 5 : Protocole sur canal avec délai pour calculer $G_{i=1}^{n} f(x[i], y[i])$

1 **si** $d \leq \frac{n}{2}$ **alors**

2 Alice et Bob effectuent le protocole 4;

3 **sinon**

4 Alice et Bob effectuent le protocole 3;

2.3 Relation avec la complexité de la communication standard

La complexité de la communication standard est beaucoup étudiée dans la littérature et elle est exactement caractérisée pour beaucoup de fonctions par des bornes inférieures et supérieures d'ordre défini. Il est intéressant de se demander comment le modèle avec délai s'y compare.

Notons qu'il est toujours possible de simuler un protocole sur un canal avec délai par un protocole de communication standard,

Soit $\mathscr{P}$ un protocole sur canal avec délai qui calcule la fonction $f(x,y)$ où $x \in \{0,1\}^n$ est l'entrée d'Alice et $y \in \{0,1\}^m$ est l'entrée de Bob.

Nous notons x_i (respectivement y_i) le bit envoyé par Alice (Bob) au moment i dans le protocole $\mathscr{P}$. Ce bit est calculé à partir de x (y) et des bits reçus par Alice (Bob) jusqu'à l'instant i, soit $y_0, \ldots, y_{i-d}$ ($x_0, \ldots, x_{i-d}$). En particulier, Alice pourra calculer x_i si elle connait $y_0, \ldots, y_{i-1}$ et Bob pourra calculer y_i s'il connait $x_0, \ldots, x_{i-1}$.

Le protocole 6 calcule f sur un canal standard, en donnant à Alice tous les bits qu'elle aurait reçu dans les $d-1$ prochaines unités de temps dans $\mathscr{P}$. Il se termine quand Alice a envoyé $x_{D_{\mathscr{P}}(f,d)}$ et Bob a envoyé $y_{D_{\mathscr{P}}(f,d)}$.

Protocole 6 : Simulation d'un protocole avec délai sur un canal standard

Alice envoie $x_1, \ldots, x_d$ à Bob;
Bob envoie $y_1, \ldots, y_d$ à Alice;
Poser $i = d+1$;
tant que $i \leq D_{\mathscr{P}}(f,d) - d$ **faire**
    Poser $j = i$;
    tant que $j \leq i+d$ **faire**
        Alice calcule $x_j = \alpha_j(x, y_0, \ldots, y_{j-d-1})$;
        Bob calcule $y_j = \beta_j(y, x_0, \ldots, x_{j-d-1})$;
        Incrémenter j;
    fin
    Alice envoie $x_i, \ldots, x_{i+d}$ à Bob;
    Bob envoie $y_i, \ldots, y_{i+d}$ à Alice;
    Incrémenter i de d unités;
fin
Alice et Bob retournent $A_i(x, y_1, \ldots, y_{i-d})$ et $B_i(y, x_1, \ldots, x_{i-d})$;

Au total, ce protocole a une complexité de $2D_{\mathscr{P}}(f,d) - d$.

2.3.1 Bornes élémentaires

Une première borne élémentaire provient du fait qu'au moins $C(f)$ bits doivent être transmis entre les participants et que le canal est bidirectionnel. Nous ignorons le cas trivial où la fonction peut être calculée sans intéraction (i.e. $C(f) = 0$).

Théorème 3.

$$D(f,d) \in \Omega\left(C(f)\right) + d.$$

Démonstration. Montrons que $D(f,d) \geq \left\lceil \frac{C(f)}{2} \right\rceil + d$. Dans un intervale $\lceil C(f)/2 \rceil + d$ unités de temps, un maximum de $C(f)$ bits peut avoir été échangé. Une complexité de la

communication avec délai plus courte briserait la minimalité de $C(f)$. □

Corollaire 1. *Le protocole 4 est optimal et est exactement optimal (plus un facteur d) quand* $n > 2d$.

Nous dérivons une première borne sur la complexité de la communication sur un canal avec délai en simulant un protocole standard.

Théorème 4.

$$D(f,d) \leq dC(f).$$

Démonstration. L'idée de la preuve est qu'on peut naïvement simuler l'exécution d'un protocole standard sur un canal avec délai en attendant d unités de temps entre chaque bit envoyé.

Nous développons un compilateur qui, sur entrée $\mathscr{P}$ un protocole de communication standard, construit un protocole $\mathscr{P}'$ pour le canal avec délai.

$\mathscr{P}$ est un arbre de hauteur $C_{\mathscr{P}}(f)$. À chaque nœud n est associé une fonction a_n qui relève d'Alice ou b_n qui relève de Bob.

Protocole 7 : Simulation d'un protocole standard sur un canal avec délai sur entrée (x,y)

1 Soit r la racine de $\mathscr{P}$;

2 À l'instant 1, Alice envoie $a_r(x)$ si elle connait la fonction de la racine de $\mathscr{P}$ et 0 sinon. Au même moment, Bob envoie $b_r(y)$ s'il connait la fonction de la racine de $\mathscr{P}$ et 0 sinon;

3 Attendre d unités de temps.

4 Au moment d, Alice et Bob connaissent le premier bit de l'exécution de $\mathscr{P}$ sur (x,y). Soit n le nœud de $\mathscr{P}$ de hauteur 1 qui correspond à cette exécution;

5 **tant que** *n n'est pas une feuille* **faire**

6 À l'instant di, Alice envoie $a_n(x)$ si c'est à elle que revient la fonction du nœud n et 0 sinon. Au même moment, Bob envoie $b_n(y)$ si c'est à lui que la fonction du nœud n et 0 sinon;

7 Attendre d unités de temps.

8 Au moment $di+d$, Alice et Bob connaissent le $(i+1)^e$ bit de l'exécution de $\mathscr{P}$ sur (x,y). On pose n le nœud de $\mathscr{P}$ de hauteur $i+1$ correspondant à cette exécution;

9 **fin**

10 Alice et Bob retournent l'étiquette de n.

Le protocole $\mathscr{P}$ prend $C_{\mathscr{P}}(f)$ bits dans le pire cas pour atteindre une feuille ce qui confère à $\mathscr{P}'$ une compexité de la communication sur un canal avec délai de $dC_{\mathscr{P}}(f)$. Si $\mathscr{P}$ est optimal, on obtient la borne recherchée. □

On remarque que le compilateur naïf laisse le canal inutilisé pour de longues périodes. Dans la section 3, nous montrons que si les participants utilisent le temps mort pour anticiper les messages futurs, on peut sauver un facteur logarithmique sur la borne du théorème 4

2.3.2 Relation avec les messages

Une façon d'exprimer un protocole sur un canal avec délai est de lister chaque message envoyé. Ceci nous donne une première borne utilisant le nombre de messages :

Théorème 5. *Tout protocole de communication $\mathscr{P}$ calculant f nous donne une borne sur $D(f,d)$. Soit $\Gamma_{\mathscr{P}}(f)$ le coût en messages de $\mathscr{P}$:*

$$D(f,d) \leq C_{\mathscr{P}}(f) + d\Gamma_{\mathscr{P}}(f) + d.$$

Démonstration. Nous simulons l'intéraction en passant un message à la fois sur le canal. Ceci nous donne l'équation :

$$\begin{aligned} D(f,d) &\leq \max_{(x,y)} \left\{ \sum_{i=0}^{\Gamma_{\mathscr{P}}(f)} (|\gamma_i(\mathscr{P},x,y)| + d) \right\} \\ &= \max_{(x,y)} \left\{ \sum_{i=0}^{\Gamma_{\mathscr{P}}(f)} |\gamma_i(\mathscr{P},x,y)| \right\} + (\Gamma_{\mathscr{P}}(f)+1)d \\ &= C_{\mathscr{P}}(f) + (\Gamma_{\mathscr{P}}(f)+1)d. \end{aligned}$$

□

Corollaire 2 (de la preuve du théorème 5)**.** *Si chaque message comporte k bits ou moins,*

$$D(f,d) \in \mathscr{O}\left(\Gamma_{\mathscr{P}}(f)(k+d)\right).$$

Si les messages sont courts, l'implémentation par message d'un protocole laisse le canal largement inutilisé. Il est tentant de dire que minimiser le nombre de messages minimisera également le temps de communication. Nous verrons dans le chapitre 4 que cette intuition ne tient pas, du moins dans le monde quantique.

2.3.3 Complexité de la communication de protocoles probabilistes sur un canal avec délai

De la même manière que dans le canal de communication standard, nous introduisons les définitions correspondantes. Encore ici, les bornes ne s'appliquent pas sur les protocoles triviaux.

Définition 25. $DR_0(f,d)$ est le coût moyen minimal d'un protocole probabiliste sur un canal avec délai qui calcule f sans erreur.

Définition 26. $DR_\varepsilon(f,d)$ est le pire coût minimal d'un protocole probabiliste sur un canal avec délai qui calcule f avec erreur ε.

Définition 27. $DR^1_\varepsilon(f,d)$ est le pire coût minimal d'un protocole probabiliste sur un canal avec délai qui calcule f avec erreur ε d'un seul côté.

En procédant par les même démonstrations des théorèmes 3 et 4 mais en utilisant les définitions probabilistes, on obtient les bornes correspondantes :

$$\begin{aligned} R_0(f)+d &\in \mathcal{O}\left(DR_0(f,d)\right) \subset \mathcal{O}\left(dR_0(f)\right) \\ R_\varepsilon(f)+d &\in \mathcal{O}\left(DR_\varepsilon(f,d)\right) \subset \mathcal{O}\left(dR_\varepsilon(f)\right) \\ R^1_\varepsilon(f)+d &\in \mathcal{O}\left(DR^1_\varepsilon(f,d)\right) \subset \mathcal{O}\left(dR^1_\varepsilon(f)\right) \end{aligned}$$

2.3.4 Complexité de la communication de protocoles quantiques sur un canal avec délai

Nous donnons la définition du volet quantique de la complexité de la communication sur un canal avec délai. À nouveau, les bornes ne s'appliquent que si une communication est nécessaire.

Définition 28. $DQ_0(f,d)$ est le coût moyen minimal d'un protocole de communication quantique sur un canal avec délai qui calcule f sans erreur.

Définition 29. $DQ_\varepsilon(f,d)$ est le pire coût minimal d'un protocole de communication quantique sur un canal avec délai qui calcule f avec erreur ε.

Définition 30. $DQ^1_\varepsilon(f,d)$ est le pire coût minimal d'un protocole de communication quantique sur un canal avec délai qui calcule f avec erreur ε d'un côté et sans erreur de l'autre côté.

$$\begin{aligned} Q_0(f)+d &\in \mathcal{O}\left(DQ_0(f,d)\right) \subset \mathcal{O}\left(dQ_0(f)\right) \\ Q_\varepsilon(f)+d &\in \mathcal{O}\left(DQ_\varepsilon(f,d)\right) \subset \mathcal{O}\left(dQ_\varepsilon(f)\right) \\ Q^1_\varepsilon(f)+d &\in \mathcal{O}\left(DQ^1_\varepsilon(f,d)\right) \subset \mathcal{O}\left(dQ^1_\varepsilon(f)\right) \end{aligned}$$

2.4 Sommaire

Nous avons introduit le modèle de complexité de la communication sur un canal avec délai. Nous avons montré comment simuler un protocole standard sur un canal avec délai et vice-versa et avons dérivé des bornes supérieures et inférieures. Nous avons étendu le modèle à un modèle de communication avec aléa et à un modèle de communcation quantique à la Cleve-Buhrman.

CHAPITRE 3

ANTICIPATION ET GAIN LOGARITHMIQUE

Le protocole donné dans la preuve du théorème 4 laisse le canal inutilisé pour de grandes périodes. En fait, seul un bit du protocole est transmis à chaque d unités de temps. Dans ce chapitre, nous montrons comment réduire cette borne par un facteur logarithmique en la taille du délai.

Théorème 6.

$$D(f,d) \in \mathscr{O}\left(\frac{dC(f)}{\lg d}\right) \quad \textit{Pour } d > 1$$

Cette nouvelle borne est particulièrement pertinente quand la communication alterne entre Alice et Bob à chaque bit.

Pour en faire la démonstration, nous introduisons la notion de forme canonique d'un protocole. Nous montrons que d'un protocole $\mathscr{P}$ dans sa forme canonique découle une borne sur la complexité de la communication sur un canal avec délai.

Lemme 3.

$$D(f,d) \leq \frac{dC_{\mathscr{P}}(f)}{\lg d - 1} \quad \textit{Pour } d > 2$$

Ce nouveau protocole sur un canal avec délai sera appelé protocole anticipatif dans les sections suivantes.

Nous expliquons ensuite comment tout protocole peut être transformé en sa forme canonique en doublant sa complexité dans le pire cas. Ceci complètera la preuve du théorème 6.

Finalement, nous montrons comment gagner un facteur 2 sur l'équation du lemme 3 pour n et d suffisamment grands.

Définition 31. On dit que l'arbre $\mathscr{P}$ représentant un protocole de communication est *dans sa forme canonique* si cet arbre binaire respecte les conditions suivantes :

- $\mathscr{P}$ est complet,
- Toutes les fonctions associées aux nœuds de hauteur impaire relèvent d'Alice et
- Toutes les fonctions associées aux nœuds de hauteur paire relèvent de Bob.

Dans un protocole sous sa forme canonique, la communication commence par Alice, puis alterne à chaque bit entre Bob et Alice. Pour toute entrée, le protocole se termine après un nombre d'échanges exactement égal à la hauteur de l'arbre. Tout protocole peut être transformé dans sa forme canonique en doublant sa complexité dans le pire cas. Nous montrerons les étapes de cette transformation à la section 3.2.

3.1 Preuve du théorème 6

Le cœur de la preuve réside dans l'idée que, dans la preuve du théorème 4, seulement un bit par d unités de temps est utile dans le calcul de $\mathscr{P}(x,y)$. Pour gagner du temps, Alice peut remplacer les bits de communication inutiles par une anticipation, i.e. le prochain bit qu'elle enverrait si Bob répondait 0 suivi de celui qu'elle enverrait si Bob répondait 1 et ainsi de suite. En usant de la même stratégie du côté de Bob, les joueurs communiquent $2\lg d$ bits de l'évaluation de $\mathscr{P}(x,y)$ à chaque $2d$ unités de temps.

Soit $\mathscr{P}$ l'arbre d'un protocole de communication sur un canal standard de hauteur $C_{\mathscr{P}}(f)$. Soient x l'entrée d'Alice et y l'entrée de Bob. Nous exigeons que $\mathscr{P}$ est dans sa forme canonique et que, sans perte de généralité, d est une puissance de 2 et $C_{\mathscr{P}}(f)$ est un multiple de $2\lg d - 1$. Nous montrons à la toute fin de la preuve comment transformer un protocole pour qu'il réponde à ces exigences.

À partir de $\mathscr{P}$, nous bâtissons un protocole $\mathscr{P}'$ sur un canal avec délai. Nous le bâtissons par blocs correspondant à des sous-arbres $\mathscr{P}_k$ de hauteur $2\lg d - 1$ de $\mathscr{P}$. La figure 3.1 est un exemple du premier de ses sous-arbres lorsque $d = 2^3$.

Dans chaque bloc, Alice et Bob relèvent les sous-arbres dont ils connaissent les fonctions et les évaluent sur leur entrée. Lors des constructions, ils utilisent les fonctions suivantes sur les arbres :

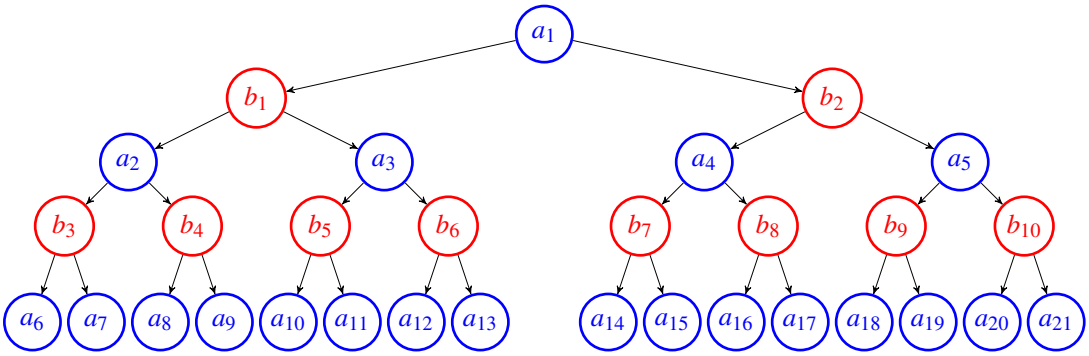


Figure 3.1 – $\mathscr{P}_1$ où Alice connait les fonctions des nœuds bleus et Bob connait les fonctions des nœuds rouges.

- $\texttt{E}(i)$ est la fonction associée au nœud i si i est un nœud interne et est l'étiquette de i si i est une feuille.
- $i.0$ est le fils gauche du nœud i.
- $i.1$ est le fils droit du nœud i.
- $R(A)$ est la racine de l'arbre A.
- $\texttt{Relever}(i,j,z)$ bâtit récursivement un sous arbre constitué seulement des nœuds relevant d'un seul participant évalués à z

$\texttt{Relever}(i,j,z)$ se comporte de la manière suivante :

Algorithme 8 : Fonction $\texttt{Relever}(i, j, z)$

Entrées : i est le nœud où on est rendu dans la construction, j est le nœud à ajouter et z est l'entrée du participant

```
1 E(i) ← E(j);
2 si j n'est pas une feuille alors
3     i.0 ← j.E(j)(z).0;
4     i.1 ← j.E(j)(z).1;
5     Relever(i.0, j.E(j)(z).0, z);
6     Relever(i.1, j.E(j)(z).1, z);
7 fin
```

Pour construire le sous-arbre de $\mathscr{P}_k$ comportant seulement les fonctions qu'elle connait, Alice effectue la construction suivante :

Algorithme 9 : Construction de $\mathscr{P}_k(A)$

```
1 P_k^A ← un nœud unique;
2 Relever(r(P_k^A), r(P_k), x);
```

Le sous arbre d'Alice est de hauteur $\lg d$. Dans notre exemple, cela correspond à l'arbre de la figure 3.2.

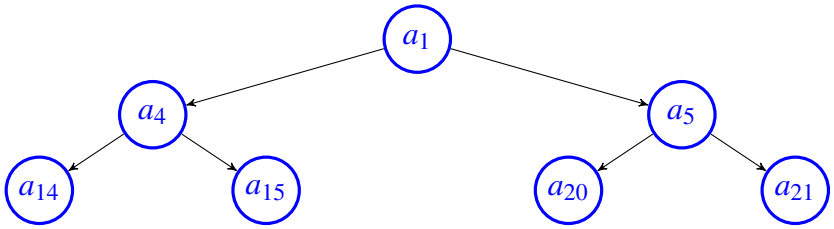


Figure 3.2 – $\mathscr{P}_1^A$ pour $a_1(x) = 1$, $a_4(x) = 0$ et $a_5(x) = 1$

Pour construire les sous-arbres de $\mathscr{P}_k$ comportant seulement les fonctions qu'il connait, Bob effectue la construction suivante :

Algorithme 10 : Construction de $\mathscr{P}_k^{B,\text{Gauche}}$ et de $\mathscr{P}_k^{B,\text{Droite}}$

1 $\mathscr{P}_k^{B,\text{Gauche}} \leftarrow$ un nœud unique;

2 `Relever`$(r(\mathscr{P}_k^{B,\text{Gauche}}), r(\mathscr{P}_k).0, y)$;

3 $\mathscr{P}_k^{B,\text{Droite}} \leftarrow$ un nœud unique;

4 `Relever`$(r(\mathscr{P}_k^{B,\text{Droite}}), r(\mathscr{P}_k).1, y)$;

Les sous-arbres de Bob sont de hauteur $\lg d - 1$. Dans notre exemple, cela correspond aux deux arbres de la figure 3.3.

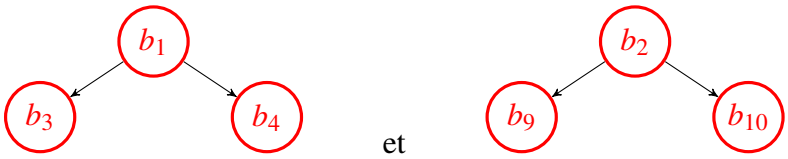


Figure 3.3 – $\mathscr{P}_1^{B,\text{Gauche}}$ et $\mathscr{P}_1^{B,\text{Droite}}$ pour $b_1(y) = 0$ et $b_2(y) = 1$

À eux trois, $\mathscr{P}_1^A, \mathscr{P}_1^{B,\text{Gauche}}$ et $\mathscr{P}_1^{B,\text{Droite}}$ couvrent complètement les $2\lg d - 1$ premiers bits de communication du protocole $\mathscr{P}$ sur entrée (x,y). Ainsi, lorsqu'il connait les 3 sous-arbres, un participant peut déterminer à quel nœud de hauteur $2\lg d - 1$ de $\mathscr{P}$ correspond une exécution de $\mathscr{P}$ sur (x,y). Nous observons aussi que les sous-arbres d'Alice et Bob peuvent se construire sans aucune connaissance de l'entrée de l'autre participant.

Décrivons maintenant $\mathscr{P}'$, un protocole sur canal avec délai qui est bâti à partir de $\mathscr{P}$ un protocol standard et qui aura la complexité voulue.

Protocole 11 : $\mathscr{P}'$ sur entrée (x,y)

1 Alice et Bob posent $\mathscr{P}_1$ égal aux $2\lg d-1$ premiers étages de $\mathscr{P}$ et $i=1$;

2 Alice bâtit $\mathscr{P}_i^A$, l'évalue sur son entrée x et l'envoie à Bob. Ceci consomme $2^{\lg d}-1=d-1$ unités de temps;

3 Au même moment, Bob bâtit $\mathscr{P}_i^{B,\text{Gauche}}$ et $\mathscr{P}_i^{B,\text{Droite}}$, les évalue sur y et les envoie à Alice. Ceci consomme $2\times(2^{\lg d-1}-1)=d-2$ unités de temps;

4 Alice et Bob attendent d unités de temps pour connaître les trois sous-arbres;

5 À partir de ces trois sous-arbres, ils déterminent chacun de leur côté à quel nœud de hauteur $2i(\lg d+1)$ de $\mathscr{P}$ correspond une exécution de $\mathscr{P}$ sur (x,y). Soit r_{i+1} ce nœud;

6 **si** r_{i+1} *est une feuille* **alors**

7 Le protocole termine et sa sortie est donnée par $\mathtt{E}(r_{i+1})$;

8 **sinon**

9 Ils posent $\mathscr{P}_{i+1}$ le sous-arbre de $\mathscr{P}$ de racine r_{i+1} et de hauteur $2\lg d-1$. Ils recommencent à l'étape 2 avec $i=i+1$ et où Alice et Bob inversent leur rôle;

10 **fin**

Au total, les étapes consomment $2d-1$ unités de temps et permettent de descendre de $2\lg d-1$ bits dans $\mathscr{P}$. Au total, nous avons bien que

$$D_{\mathscr{P}'}(f,d)=(2d-1)\times\frac{C_{\mathscr{P}}(f)}{2\lg d-1}$$

Reste à gérer les cas limites où d n'est pas une puissance de 2 ou quand $C_{\mathscr{P}}(f)$ n'est pas un multiple de $2\lg d-1$.

Si d n'est pas une puissance de 2, nous divisons simplement $\mathscr{P}$ en section de hauteur $2\lfloor\lg d\rfloor-1$ afin d'obtenir la borne $D(f,d)\leq 2\times d\times\frac{C_{\mathscr{P}}(f)}{2\lfloor\lg d\rfloor-1}$.

Finalement, si $C_{\mathscr{P}}(f)$ n'est pas un multiple de $2\lfloor\lg d\rfloor-1$, nous pouvons ajouter des étages à $\mathscr{P}$ jusqu'à ce qu'on atteigne le prochain multiple. Nous obtenons la nouvelle

borne $D(f,d) \leq 2 \times d \times \left\lceil \frac{C_{\mathscr{P}}(f)}{2\lfloor \lg d \rfloor - 1} \right\rceil$.

3.2 Transformation d'un protocole dans sa forme canonique

Nous montrons maintenant comment bâtir $\mathscr{P}'$ pour un protocole n'étant pas un arbre binaire complet, dans lequel l'intéraction n'alterne pas entre Alice et Bob, ou dans lequel Bob parle en premier. Chacun de leur côté, Alice et Bob effectuent les opérations suivantes sur $\mathscr{P}$:

3.2.1 Étape 1 - Forcer l'intéraction

On effectue une fouille en largeur et quand un sommet n est étiqueté par une fonction relevant de Bob, mais que les nœuds de l'étage doivent relever d'Alice, on remplace ce sommet par un nœud relevant d'Alice de fonction qui retourne zéro sur toute entrée et on place deux copies du sous-arbre de racine n chez les deux fils du nouveau nœud, comme illustré dans la figure 3.4

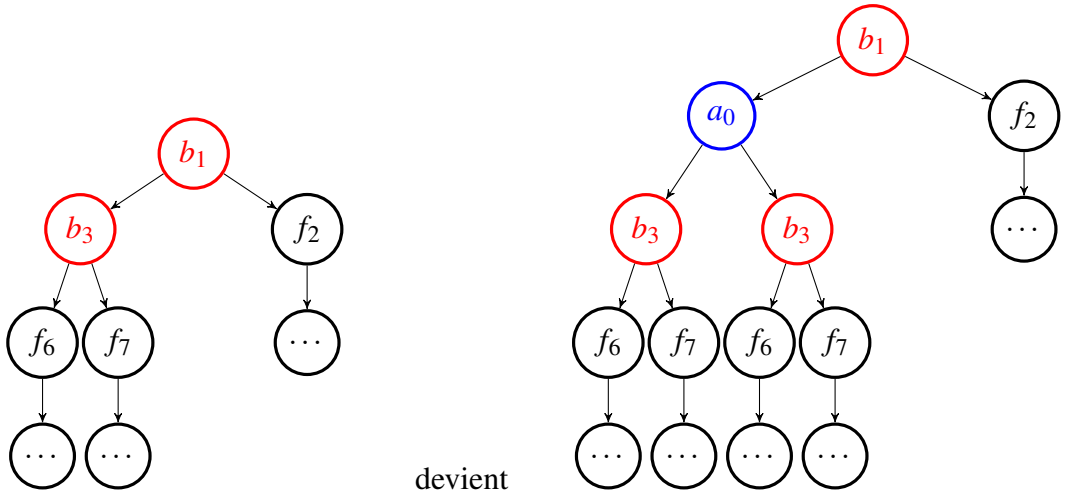


Figure 3.4 – Comment forcer l'intéraction entre Alice et Bob

Forcer l'intervention de Bob est analogue. À chaque étage du protocole, nous ajoutons au maximum un étage pour forcer l'intéraction. La hauteur de l'arbre va doubler dans le pire cas.

3.2.2 Étape 2 - Descendre les feuilles

Après avoir forcé une alternation entre les participants, nous descendons les feuilles au niveau le plus bas. Pour ce faire, nous effectuons une fouille en largeur et à chaque fois que nous trouvons une feuille f qui n'est pas au niveau le plus bas, nous la remplaçons par un nœud avec la fonction qui retourne toujours zéro relevant du participant dont c'est le tour de parler et nous mettons deux copies f sous ce nouveau nœud. Nous illustrons cette étape dans la figure 3.5.

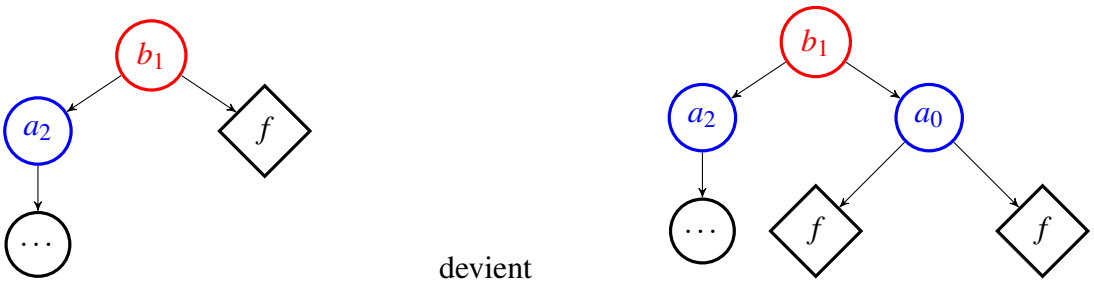


Figure 3.5 – Comment descendre une feuille

Après avoir descendu toutes les feuilles au niveau le plus bas, nous nous retrouvons avec un arbre binaire complet. La hauteur de l'arbre reste la même.

3.2.3 Étape 3 - Commencer par Alice

Finalement, si c'est Bob qui commence la communication dans $\mathscr{P}$, on place deux copies de $\mathscr{P}$ sous une nouvelle racine de fonction zéro associée à Alice, comme illustré dans la figure 3.6.

Cette étape n'ajoute qu'un étage à la hauteur de $\mathscr{P}$.

3.3 Gain logarithmique double

Nous observons que dans l'exécution $\mathscr{P}'$, le canal est inutilisé la moitié du temps. Si les participants encodent leurs arbres dans l'ordre d'une fouille en largeur de $\mathscr{P}$, la dernière moitié de la chaîne qu'ils envoient ne contient que le dernier niveau, et donc qu'un

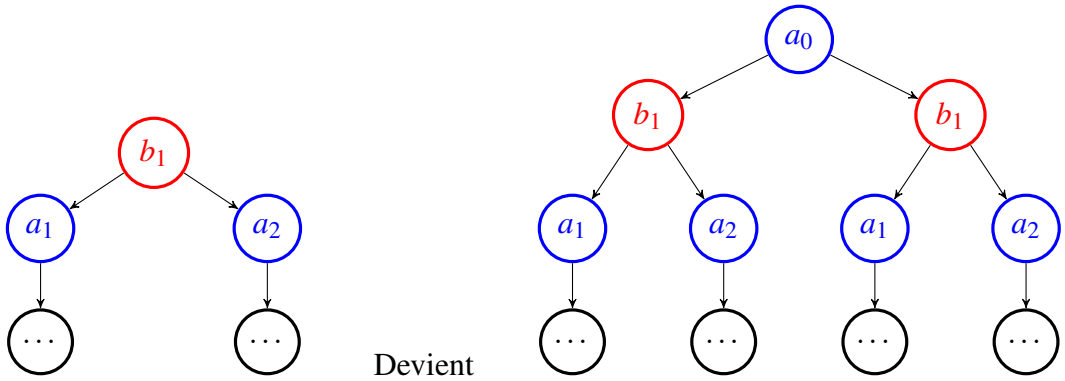


Figure 3.6 – Comment commencer par Alice

seul bit de l'exécution de $\mathscr{P}$. Les participants peuvent anticiper la suite de l'exécution plus tôt, cette fois-ci en ayant deux racines possibles.

On peut appliquer la même idée, mais cette fois-ci pour anticiper après $d/2^k$ unités de temps. À ce moment, les $2\lg d - 1 - 2k$ premiers étages de $\mathscr{P}$ sont connus par les deux participants. Ils peuvent décrire les 2^k arbres commençant à l'étage $2\lg d - 1$ et de racine correspondant à l'exécution de $\mathscr{P}$ sur (x,y). Afin que l'anticipation procure un gain, la description des 2^k arbres doit être contenue dans d unités de temps.

Pour chaque k tel que $k2^k \leq d$, la description du protocole $\mathscr{P}^k$ correspond à cette stratégie d'anticipation.

Pour faciliter la lecture, nous séparons le protocole en deux parties. Dans la partie préliminaire, soit le protocole 12, Alice et Bob envoient respectivement les $\lg d$ et $\lg d - 1$ premiers étages de $\mathscr{P}$ qui leur revient en ordre de fouille en largeur. Au moment $d + d/2^k$, ils anticipent les k prochains bit de l'exécution de $\mathscr{P}$. Nous donnons d'abord le protocole et illustrons ensuite les informations qu'ils partagent sur $\mathscr{P}$ étape par étape.

Protocole 12 : Préliminaires du protocole $\mathscr{P}^k$ sur entrée (x,y)

1 Alice envoie les $\lg d$ premiers étages de $\mathscr{P}$ qu'elle connait en ordre de fouille en largeur. Au même moment, Bob envoie les $\lg d - 1$ premiers étages de $\mathscr{P}$ qu'il connait en ordre de fouille en largeur. Ce qui consomme moins de d unités de temps;

2 Alice et Bob attendent $d/2^k$ unités de temps pour connaître les $2(\lg d - k) - 1$ premiers étages de $\mathscr{P}$;

3 Ils déterminent chacun de leur côté auxquels 2^k arbres dont la racine a une hauteur de $2\lg d - 1$ de $\mathscr{P}$ correspondent une exécution de $\mathscr{P}$ sur (x,y). Soient $\mathscr{P}_1$ à $\mathscr{P}_{2^k}$ ces sous-arbres (voir figure 3.7);

4 Alice envoie les $\lg d - k - 1$ premiers étages des 2^k arbres $\mathscr{P}_1, \mathscr{P}_2, \ldots, \mathscr{P}_{2^k}$, qu'elle connait. Elle décrit leurs nœuds en ordre de fouille en largeur dans $\mathscr{P}$. Au même moment, Bob envoie les $\lg d - k$ premiers étages des 2^k arbres $\mathscr{P}_1, \mathscr{P}_2, \ldots, \mathscr{P}_{2^k}$ qu'il connait. Il décrit leurs nœuds en ordre de fouille en largeur dans $\mathscr{P}$. Ceci consomme d unités de temps. En même temps, ils reçoivent les k derniers étages de $\mathscr{P}$ qui ont été envoyés à l'étape 1;

5 Ils déterminent chacun de leur côté auquel nœud de hauteur $2\lg d - 1$ correspond l'exécution de $\mathscr{P}$ sur (x,y) (voir figure 3.8);

À la fin du protocole préliminaire, Alice et Bob partagent $2\lg d - 1$ étages de $\mathscr{P}$ et ont utilisé $2d + d/2^k$ unités de temps. Ils ont toute l'information nécessaire pour déterminer lequel des 2^k sous arbres envoyés contient l'exécution de $\mathscr{P}$ sur (x,y).

On illustre dans la figure 3.7 le parcours de $\mathscr{P}$ qu'Alice et Bob effectuent au cours de l'étape 1 à 3 du protocole 12. Le chemin vert représente les bits de l'exécution de $\mathscr{P}$ sur (x,y) qu'Alice et Bob connaissent. Ceux en pointillés représentent les k chemins possibles qu'ils anticipent à l'aide des arbres $\mathscr{P}_1$ à $\mathscr{P}_k$, ici en blanc.

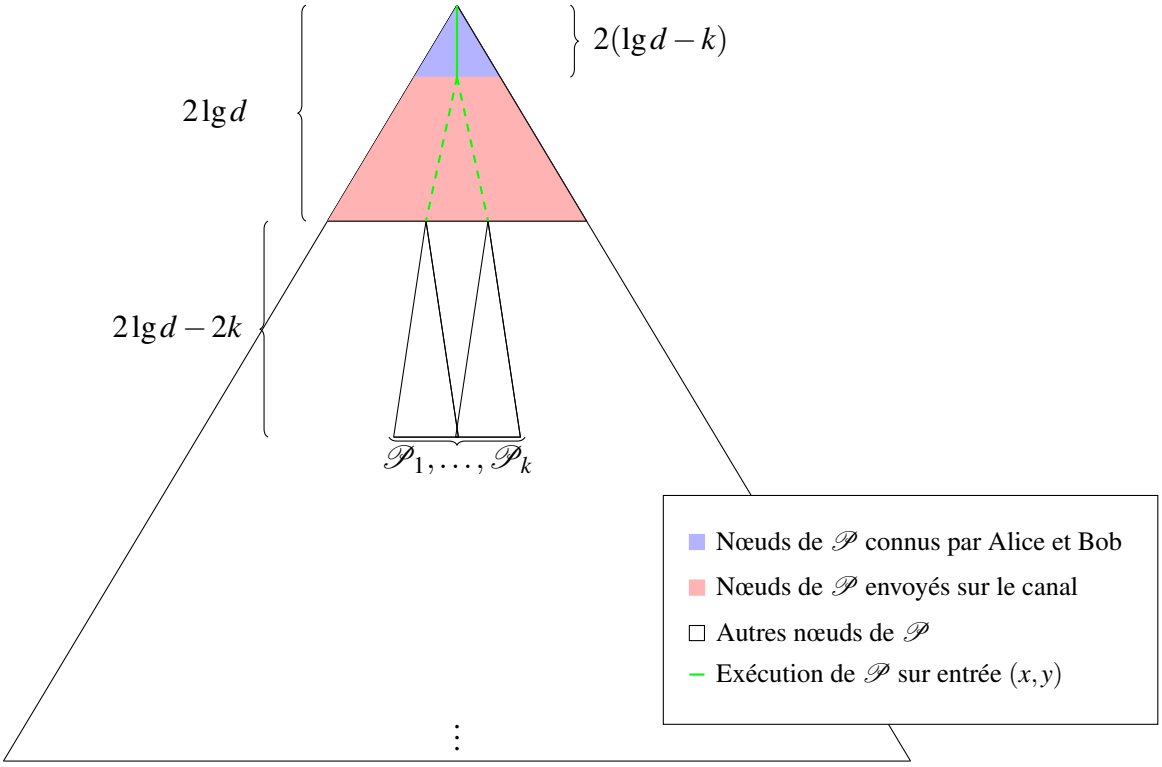


Figure 3.7 – État de l'exécution de $\mathscr{P}$ après l'étape 3 du protocole 12

Dans la figure 3.8, illustrant la fin du protocole 12, Alice et Bob ont envoyé les premiers étages des 2^k arbres suivant le nœud de hauteur de $2\lg d - 1$ correspondant à une exécution de $\mathscr{P}$ sur (x,y).

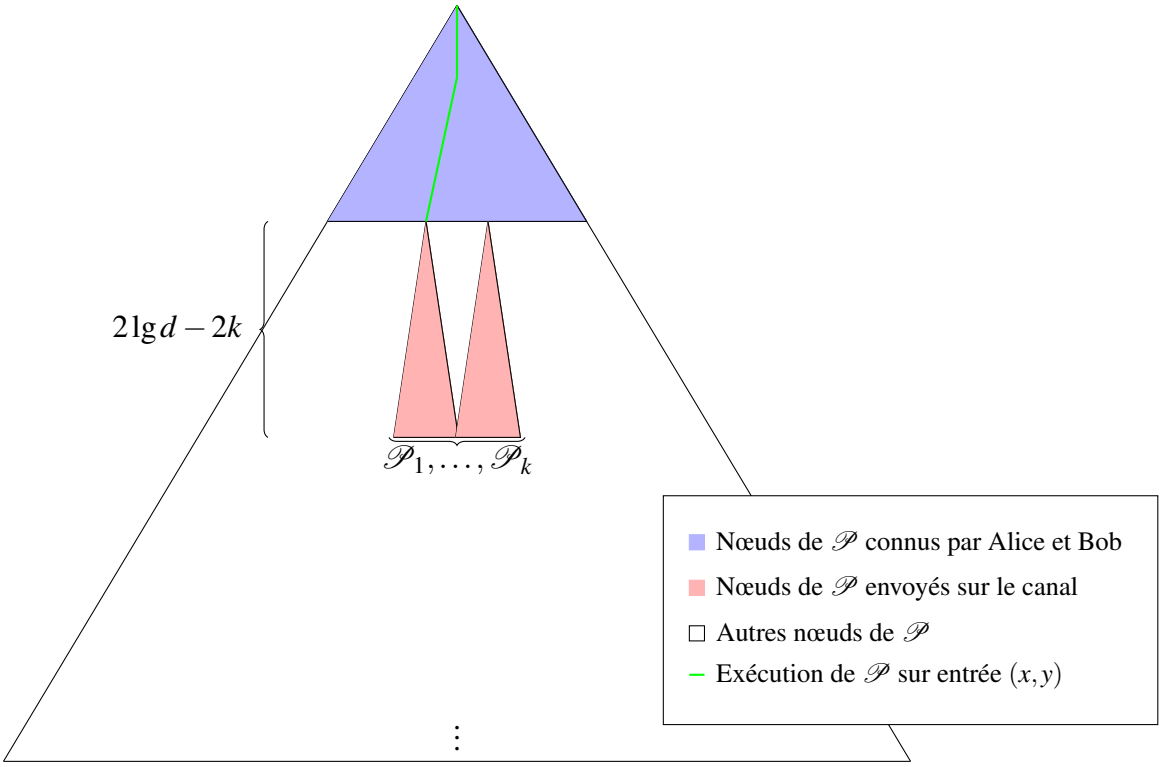


Figure 3.8 – État de l'exécution de $\mathscr{P}$ à la fin du protocole 12

Après le protocole préliminaire, Alice et Bob attendent $d/2^k$ unités de temps, soit après avoir reçu $\lg \frac{d/2^k}{2^k} = \lg d - 2k$ étages de chacun des 2^k arbres. Alice et Bob poursuivent en échangeant respectivement $\lg d - 2k - 1$ et $\lg d - 2k$ étages de 2^k arbres anticipés de $\mathscr{P}$. La description de $\lg d - 2k$ étages de 2^k arbres prend $2^k \times 2^{\lg d - k} = d$ unités de temps. Ils poursuivent cette stratégie jusqu'à ce qu'ils atteignent une feuille de $\mathscr{P}$. À chaque fois, il peuvent déterminer $2\lg d - 4k - 1$ nouveaux bits de l'exécution de $\mathscr{P}$ sur (x,y) et utilisent $d + d/2^k$ unités de temps.

Protocole 13 : $\mathscr{P}^k$ sur entrée (x,y)

1 Alice et Bob effectuent le protocole 12. Ils posent $i = 2\lg d + 2k - 1$;

2 Alice et Bob attendent $d/2^k$ unités de temps pour connaître les $i + 2(\lg d - 2k) - 1$ premiers étages de $\mathscr{P}$. Ils posent $i := i + 2(\lg d - 2k) - 1$;

3 **si** $\mathscr{P}$ *a une hauteur inférieure à* $i + 2k$ **alors**

4 Alice et Bob attendent d unités de temps afin de connaître toute l'exécution de $\mathscr{P}$ sur (x,y). Le protocole se termine et sa sortie est donnée par l'étiquette de la feuille atteinte par cette exécution;

5 **sinon**

6 Ils déterminent chacun de leur côté auxquels 2^k arbres dont la racine est de hauteur $i + 2k$ de $\mathscr{P}$ correspondent une exécution de $\mathscr{P}$ sur (x,y). Soient $\mathscr{P}_1$ à $\mathscr{P}_{2^k}$ ces sous-arbres;

7 Alice envoie les $\lg d - k - 1$ premiers étages des 2^k arbres $\mathscr{P}_1, \mathscr{P}_2, \ldots, \mathscr{P}_{2^k}$, qu'elle connait. Elle décrit leurs nœuds en ordre de fouille en largeur dans $\mathscr{P}$. Au même moment, Bob envoie les $\lg d - k$ premiers étages des 2^k arbres $\mathscr{P}_1, \mathscr{P}_2, \ldots, \mathscr{P}_{2^k}$ qu'il connait. Il décrit leurs nœuds en ordre de fouille en largeur dans $\mathscr{P}$. Ceci consomme d unités de temps (voir la figure 3.9);

8 Pendant ce temps, ils reçoivent $k/2$ étages de $\mathscr{P}$ envoyés à l'étape 4 du protocole préliminaire (ou de l'étape 7 du tour de boucle précédent);

9 Ils déterminent chacun de leur côté auquel nœud de hauteur $i + 2k$ correspond l'exécution de $\mathscr{P}$ sur (x,y) (voir la figure 3.10);

10 Alice et Bob recommencent à l'étape 2 où Alice et Bob inversent leur rôle;

11 **fin**

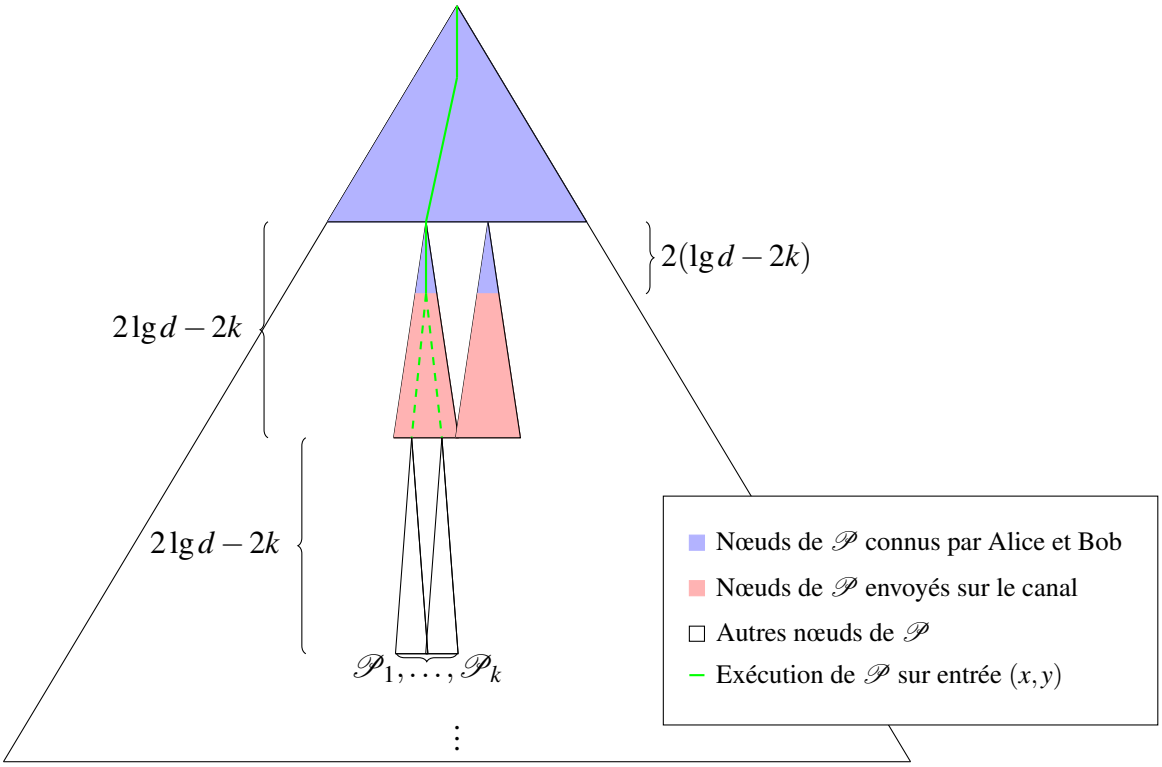


Figure 3.9 – État de l'exécution de $\mathscr{P}$ à la fin de l'étape 7 du premier tour de boucle du protocole 13

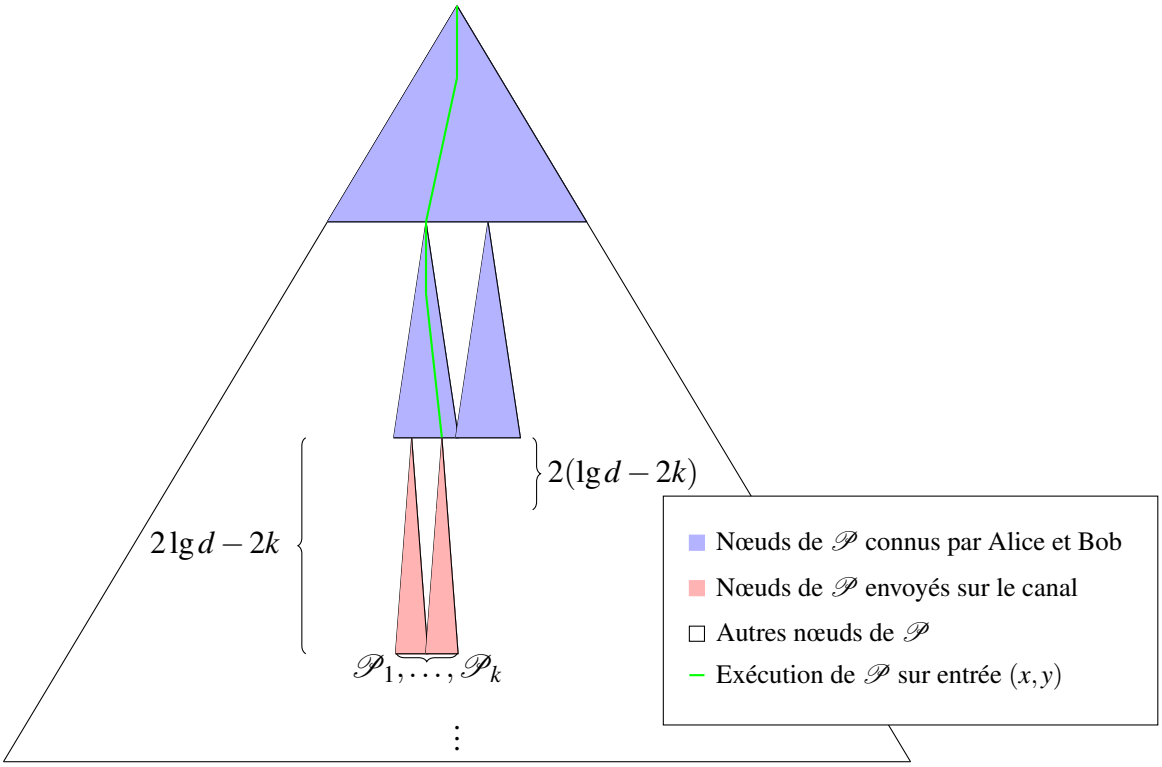


Figure 3.10 – État de l'exécution de $\mathscr{P}$ à la fin de l'étape 9 du premier tour de boucle du protocole 13

Au total, à chaque $d + d/2^k$ unités de temps, Alice et Bob descendent de $2(\lg d - 2k) - 1$ étages dans $\mathscr{P}$. Après avoir pris en compte le protocole préliminaire, on obtient pour tout $k2^k \leq d$ l'équation :

$$D_{\mathscr{P}^k}(f,d) = d\left(1+\frac{1}{2^k}\right)\frac{C_{\mathscr{P}}(f) - 2\lg d - 1}{2(\lg d - 2k) - 1} + 2d + \frac{d}{2^k} \approx \frac{\left(1+\frac{1}{2^k}\right)}{2}\frac{dC_{\mathscr{P}}(f)}{\lg d - 2k}$$

Constituant un protocole deux fois plus rapide que le protocole 10.

CHAPITRE 4

SOLUTIONS EXPLOITANT LE DÉLAI

Un facteur multiplicatif de $d/\lg d$ sur la complexité de la communication standard est encore très loin de la borne inférieure élémentaire de $C(f)+d$.

Un premier gain sur la borne donnée par le théorème 6 est fait lorsque la fonction à calculer est parallélisable. Dans le meilleur cas, la fonction peut être parallélisable en d sous-protocoles, comme on verra à l'aide de l'équation 4.1. Dans ce cas-ci, on peut gagner un facteur multiplicatif de $d/\lg d$ sur la complexité de la communication sur un canal avec délai de l'implémentation anticipative et,alternativement, un facteur multiplicatif d sur la complexité de la communication sur un canal avec délai de l'implémentation naïve.

Nous présentons d'abord un exemple ou un protocole optimal calculant une fonction de type *somme directe* a avantage à être parallélisée dans le modèle de complexité de la communication sur un canal avec délai. Dans ce cas-ci, l'implémentation naïve d'un protocole optimal est avantageuse sur une implémentation anticipative.

Puis nous donnons un exemple de protocole quantique optimal en terme de complexité de la communication qui fournit des protocoles quantiques anticipatifs et naïfs sous-optimaux en terme de complexité de la communication sur un canal avec délai.

4.1 Exemple de gain dans un protocole optimal

Nous introduisons une fonction pour laquelle l'implémentation anticipative d'un protocole optimal en terme de complexité de la communication est sous-optimale sur un canal avec délai.

Définition 32. [30] Soit un arbre binaire complet de hauteur k et soient deux participants Alice et Bob. Alice a comme entrée x un étiquetage binaire des nœuds de niveau pair et

Bob a comme entrée y un étiquetage binaire des nœuds de niveau impair. L'étiquette d'un nœud interne représente un pointeur vers un de ses fils. À chaque étiquetage est donc associé un chemin unique de la racine vers une feuille. Le *problème d'arbre*, noté $T_k(x,y)$, cherche à déterminer l'étiquette de cette feuille à partir des entrées.

Exemple 5. L'exemple 4.1 est une instance du problème d'arbre de hauteur 3 où Alice connaît les bits en bleu et Bob connait les bits en rouge. La sortie de cet exemple est 1, soit la troisième feuille, et le chemin effectué par Alice et Bob est en gras.

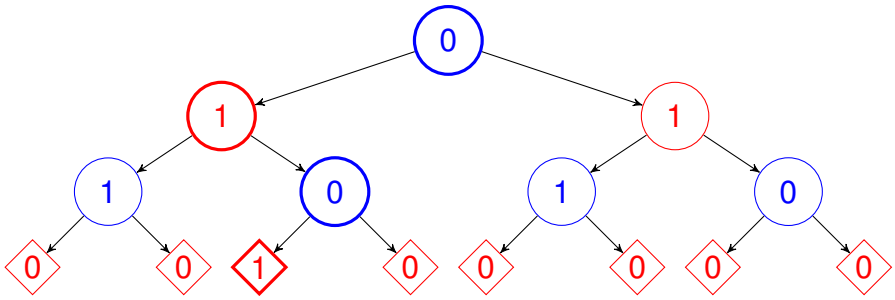


Figure 4.1 – Instance du problème d'arbre

Lemme 4.

$$C(T_k) \in \Theta(k)$$

Démonstration. Étudions la sortie de T_k sur les arbres où les feuilles sont toutes étiquetées par 0 sauf une. Soit f_1 cette feuille. Exactement un chemin se rend à f_1 à partir de la racine. La paire d'entrée qui atteint cette feuille aura 1 comme valeur dans la matrice de T_k. Par contre, toutes les autres cases de la rangée seront nulles puisqu'elles sont associées à un chemin qui ne termine pas à f_1. Nous avons donc que la matrice associée à T_k contient la matrice identité de taille 2^{k-1}. Comme $C(T_k) \geq \lg(\text{rang}(T_k))$, nous avons $C(T_k) \in \Omega\left(\lg 2^{k-1}\right)$

Nous montrons que cette borne est optimale en donnant un protocole $\mathscr{P}_{\text{Arbre}}$ de complexité $k+1$: Sur entrée (x,y), Alice et Bob donnent chacun leur tour l'étiquette du nœud auquel ils sont rendus à commencer par la racine jusqu'à la feuille. Au total, ils communiquent un bit de plus que la hauteur de l'arbre. □

Étudions à maintenant une version différente du problème d'arbre. Dans celui-ci, le but d'Alice et Bob est de calculer le $\wedge$ de d problèmes d'arbres sur leurs entrées $x_1, \ldots, x_d$ et $y_1, \ldots, y_d$. Nous le notons T_k^d puisqu'il s'agit d'un problème de type somme directe.

Lemme 5.

$$C(T_k^d) \in \Theta(dk)$$

Démonstration. D'abord, nous avons vu dans le chapitre préliminaire que pour les problème de somme directe, $C(T_k^d) \in \mathscr{O}(dk)$. Aussi, nous avons vu que $\text{rang}(f \wedge g) = \text{rang}(f) \cdot \text{rang}(g)$. Nous avons donc que

$$C(T_n^d) \geq \lg\left(\text{rang}\left(\bigwedge_{i=1}^{d} T_n(x_i, y_i)\right)\right) = \lg\left(\text{rang}(T_k)^d\right) = dk$$

Nous pouvons atteindre cette borne en effectuant le protocole $\mathscr{P}_{\text{Arbre}}$ sur les d arbres un à la suite de l'autre et en calculant le $\wedge$ des feuilles. □

Si nous ne faisons que simuler le protocole $\mathscr{P}_{\text{Arbre}}$ d fois avec la méthode par anticipation vue dans le chapitre 3, puis calculons le $\wedge$ sur les feuilles, on obtient une borne supérieure sur la complexité de la communication sur un canal avec délai de

$$D(T_k^d, d) \in \mathscr{O}\left(d^2 k / \lg d\right).$$

Or, il est possible de calculer T_k^d beaucoup plus rapidement en disposant du délai intelligemment et en résolvant les d problèmes d'arbres en parallèle, ce qui nous donne l'équation :

$$D(T_k^d, d) \in \mathscr{O}(dk) \tag{4.1}$$

On observe qu'un autre protocole optimal sur canal standard, celui dans lequel chaque problème d'arbre est fait en parallèle, lorsqu'implanté naïvement sur le canal avec délai, aurait été optimal. Nous sommes tentés de dire qu'un protocole optimal avec un nombre minimal de messages est toujours optimal lorsqu'implémenté naïvement sur un

canal avec délai. Cette intuition est fausse, comme on verra dans l'exemple quantique de la section suivante. Par contre, nous ne connaissons aucun contre exemple de cette intuition dans le monde classique.

4.2 Exemple de gain dans un protocole quantique optimal

Nous montrons qu'une utilisation astucieuse du délai pour répondre au problème d'ensembles disjoints est plus efficace qu'une implémentation naïve et anticipative d'un protocole quantique optimal sur un canal standard.

Le problème d'ensembles disjoints, tel qu'introduit à la section 2.2, a une complexité de la communication quantique égale à $Q(\texttt{ENSEMBLES\_DISJOINTS}) = \Theta(\sqrt{n})$.

Alexander Razborov a prouvé que $\sqrt{n}$ constitue une borne inférieure dans [38] ce qui est quadratiquement mieux que la borne inférieure classique dans $\Omega(n)$. Puis Harry Buhrman, Richard Cleve et Avi Wigderson. donnent un premier protocole quasi-optimal d'ordre $\mathscr{O}(\lg n\sqrt{n})$ dans [8], le protocole est amélioré par Peter Høyer et Ronald de Wolf dans [19], puis par Scott Aaronson et Andris Ambainis dans [1] où on arrive enfin à l'optimalité.

La stratégie de ces trois protocoles est d'appliquer l'algorithme de Grover [18],[17]. Nous ne donnerons pas les protocoles en détail, mais nous soulignons que chacun de ceux-ci nécessitent beaucoup d'intéraction de par leur utilisation de l'algoritme de Grover ayant $\sqrt{n}$ messages.

Si nous simulons naïvement, par le théorème 5, le protocole de Scott Aaronson et Andris Ambainis sur un protocole avec délai, nous obtenons la borne suivante :

$$\mathrm{DQ}(\texttt{ENSEMBLES\_DISJOINTS}, d) \in \mathscr{O}\left(d\sqrt{n}\right)$$

En utilisant la stratégie anticipative, on obtient la borne

$$\mathrm{DQ}(\texttt{ENSEMBLES\_DISJOINTS}, d) \in \mathscr{O}\left(d\sqrt{n}/\lg d\right).$$

Il est possible de faire mieux sur un canal avec délai. Plutôt que d'effectuer l'algorithme d'Aaronson et Ambainis sur toute l'entrée de taille n, ce qui coûte $\mathcal{O}(1) \times \sqrt{n}$ rondes de communication chacun ayant une taille constante, nous l'exécutons en parallèle sur des sections de l'entrée.

Soit k la taille maximale d'une ronde de communication dans l'algorithme de Scott Aaronson et Andris Ambainis et $\ell\sqrt{n}$ le nombre de rondes de communication. Le protocole 14 calcule également le problème d'ensembles disjoints.

Protocole 14 : Algorithme d'Aaronson et Ambainis en parallèle

1. Alice et Bob effectuent les étapes de chacune des $\ell\sqrt{n}$ itérations constituant l'algorithme de Scott Aaronson et Andris Ambainis en parallèle sur d/k problèmes d'ensembles disjoints de taille nk/d;
2. Alice et Bob calculent chacun de leur côté le $\vee$ des sorties des nk/d problèmes d'ensembles disjoints et le retournent;

Ce protocole est consitué de $\ell\sqrt{nk/d}$ rondes de taille d. Si nous l'implémentons sur un canal avec délai, nous avons une complexité de $\mathcal{O}\left(d\sqrt{n/d}\right)$.

Reste à trouver sa probabilité de succès, soit la probabilité que protocole ait réussi simultanément les d/k sous-problèmes. Selon Michel Boyer, Gilles Brassard, Peter Høyer et Alain Tapp [4], l'algorithme de Grover a une probabilité de succès $(n-1)/n$ après $\pi\sqrt{n}/4$ itérations. En ajoutant $\pi/4$ à ℓ, le protocole 14 a une probabilité de succès de

$$\left(\frac{nk/d-1}{nk/d}\right)^d \geq \left(\frac{d^2-1}{d^2}\right)^d > \frac{2}{3}$$

Dès que $n \geq d^3/k$ et que $d > 3$.

Nous avons donc la nouvelle borne pour $n \geq d^3$:

$$\mathrm{DQ}(\texttt{ENSEMBLES\_DISJOINTS}, d) \in \mathcal{O}\left(\sqrt{dn}\right).$$

Cette utilisation du délai offre un avantage $\sqrt{d}$ par rapport à l'implémentation naïve de l'algorithme de Scott Aaronson et Andris Ambainis. Elle est aussi plus efficace son implémentation anticipative sur un canal avec délai, offrant un avantage de $\sqrt{d}/\lg d$ sur celle-ci

4.2.1 Observations sur le nombre de messages

L'algorithme de Grover est considéré comme optimal en terme de nombre de messages, bien que les meilleurs résultats ne fournissent qu'une borne inférieure de

$\Gamma(\texttt{ENSEMBLES\_DISJOINTS}) \in \Omega\left(n^{1/4}\right)$ ([21],[29]). Si on considère que le protocole de Scott Aaronson et Andris Ambainis est minimal en terme de messages, nous avons présenté un exemple de protocole optimal et de quantité de messages minimale pour lequel une implémentation naïve sur un canal avec délai est sous-optimale.

4.3 Sommaire

Nous avons donnés deux exemples dans lesquel une implémentation anticipative d'un protocole optimal sur un canal standard n'est pas optimal sur un canal avec délai. Dans le cas quantique, même un protocole optimal à la fois en terme de complexité de la communication et en terme de nombre de messages ne garantit pas une implémentation naïve optimale sur un canal avec délai.

CHAPITRE 5

APPLICATION CRYPTOGRAPHIQUES

Jusqu'à présent les tâches introduites sont des tâches de communication sans saveur cryptographique. Nous introduisons certaines tâches cryptographiques de base. Nous résolvons certaines d'entre elles à partir de la promesse que le canal de communication entre les participants est muni d'un délai.

5.1 Échange de bit

Une première primitive cryptographique est l'échange de bit. Cette primitive permet à Alice et Bob d'échanger leurs bits a et b avec la garantie qu'ils sont indépendants. Par exemple Alice ne pourra pas modifier son bit a après avoir vu celui de Bob.

Nous représentons un échange de bit sous forme de boîte noire.

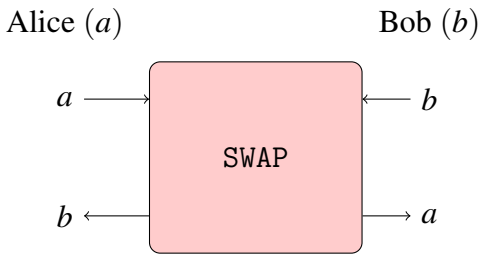


Figure 5.1 – Échange de bit

À partir d'un procédé d'échange de bit, Alice et Bob peuvent effectuer une première tâche cryptographique, celle du pile ou face à distance introduite par Manuel Blum dans [3].

Alice et Bob veulent tirer à pile ou face par téléphone. Bob doit pouvoir choisir à quel participant correspond chaque côté de la pièce et Alice doit choisir lequel des côtés est gagnant. La difficulté de cette tâche est que les participants doivent être convaincus que

l'autre ne triche pas. Par exemple, si c'est Bob qui fait le tir, il doit être assuré qu'Alice ne changera pas son choix de après avoir connu le résultat.

Le tir est très facile à effectuer à partir d'un échange de bit.

Protocole 15 : Pile ou face à partir d'un échange de bit

1 Alice et Bob s'entendent sur un encodage binaire des côtés de la pièce;
2 Ensuite, Alice et Bob envoient leur choix à `SWAP`;
3 Ils déterminent qui a gagné;

Lorsqu'Alice et Bob ont accès à un canal avec délai, ils peuvent simuler une boîte `SWAP`. Nous soulignons qu'il est essentiel que le canal ait un délai connu d'au moins d, sinon tout avantage cryptographique est perdu.

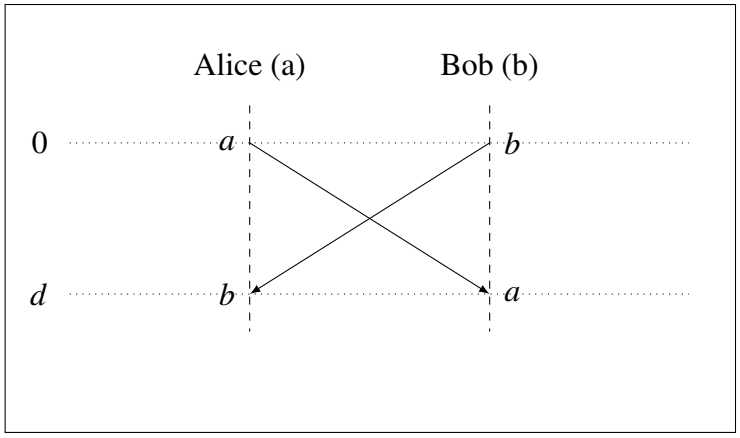


Figure 5.2 – Échange de bit utilisant le délai

Si on enlève l'assurance qu'Alice et Bob communiquent à travers un canal avec délai d, la promesse que l'information ne peut pas voyager plus vite la lumière est également suffisante pour réaliser un échange de bit cryptographique. Plutôt que d'utiliser un canal de délai d, Alice et Bob s'entendent sur deux positions séparées par une distance qui sera parcourue en d unités de temps. Alice se place à la première position et Bob à la deuxième. Pour communiquer, ils émettent leur bit dans toutes les directions (voir la figure 5.3).

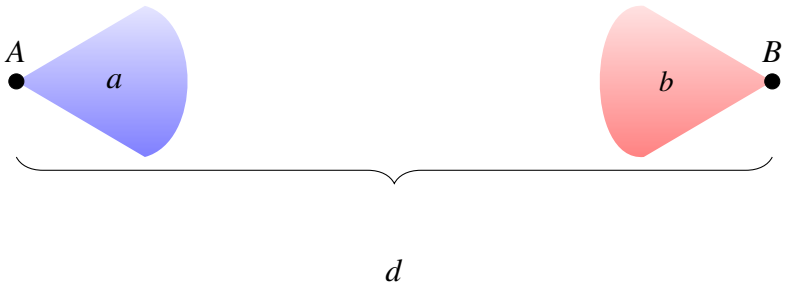


Figure 5.3 – Configuration d'Alice et Bob pour l'échange de bit sans canal

Dans ces conditions, le protocole de la figure 5.2 n'est plus sécuritaire. Nous utilisons la tactique utilisée dans [9] pour établir une stratégie d'attaque par Bob contre Alice honnête dans la configuration 5.3. Si Bob se place très près d'Alice, il reçoit a très vite et choisit b en conséquence. Il attend le temps nécessaire à un bit de traverser la distance entre sa supposée position et la position d'Alice avant d'envoyer son bit. Alice reçoit b au moment prévu.

Protocole 16 : Attaque contre le protocole 5.2 dans le modèle sans canal.

1. Alice se place à la position prévue mais Bob se place à une petite distance ε d'Alice;
2. À l'instant 0, Alice émet le bit a;
3. À l'instant ε, Bob reçoit a et choisit b en conséquence;
4. Bob attend $d - 2\varepsilon$ unités de temps et émet b;
5. Alice reçoit b à l'instant d;

Pour contrer cette attaque, nous présentons une version simplifiée de la méthode utilisée par Adrian Kent pour réaliser sa mise en gage dans [26]. Alice place un partenaire A' à une petite distance ε de la position escomptée de Bob (voir la figure 5.4). On fait l'hypothèse qu'Alice et A' collaborent avec une confiance mutuelle complète. Si A' reçoit à l'instant ε le même bit qu'Alice reçoit à l'instant d, Alice est assurée que Bob a choisi son bit indépendemment de a. Bob place également un guêt B' près d'Alice. Évidemment

la restricition sur la vitesse de transmission de l'information s'applique aussi entre les joueurs et leur partenaire.

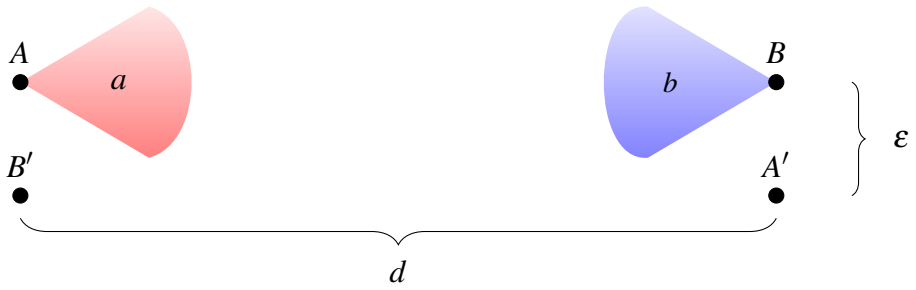


Figure 5.4 – Protocole sécuritaire sans canal pour l'échange de bit

Ceci donne le protocole d'échange de bit cryptographique sécuritaire basé sur la relativité :

Protocole 17 : Protocole relativiste d'échange de bit sécuritaire

1. Alice émet le bit a et Bob émet le bit b;
2. À l'instant ε, B' reçoit a et A' reçoit b;
3. À l'instant d, Alice reçoit b et Bob reçoit a;
4. A et A' comparent leur bit. B et B' font de même de leur côté. Cette étape prend d unité de temps;
5. Si la comparaison échoue, le protocole est avorté;

5.2 Mise en gage

La mise en gage, formellement décrite par Gilles Brassard, David Chaum et Claude Crépeau dans [7], est une primitive cryptographique constituée de deux phases. Dans la première phase, Alice s'engage à un bit a. Dans la deuxième phase, Alice peut décider de révéler a à Bob. Entre les deux phases, les participants sont libres de laisser s'écouler autant de temps qu'ils le désirent. De plus, Alice est libre de ne jamais révéler a.

Un procédé de mise en gage doit également respecter les conditions de sécurité suivantes :

- Alice ne peut pas changer le bit auquel elle est engagée,
- Bob ne connait rien de a avant la phase où il est révélé.

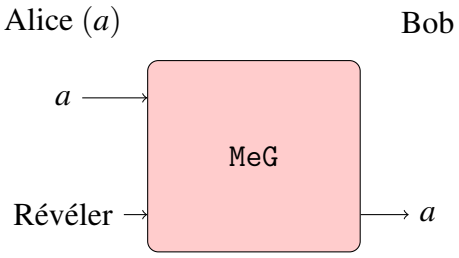


Figure 5.5 – Mise en gage

Adrian Kent a monré dans [25] que la mise en gage est strictement plus forte que l'échange de bit. En plus du pile ou face, la mise en gage permet d'effectuer de nombreuses tâches cryptographiques, comme les preuves à révélation nulles, les procédés de signatures et le partage de secret vérifiable.

À partir de la garantie que deux participants ne peuvent pas communiquer plus vite que la vitesse de la lumière, Adrian Kent développe dans [26] un procédé de pseudo mise en gage.

Dans son procédé, entre la phase d'engagement et d'ouverture, les participants doivent effectuer une phase de conservation. Dans cette phase, les participants doivent garder la mise en gage active en échangeant des messages qui grandissent exponentiellement (plus tard, Kent réduit la taille des messages de conservation à une constante [27]).

5.3 Transfert équivoque

Nous montrons mainenant que délai ne semble pas offrir de primitives cryptographiques plus puissantes, comme le transfert équivoque.

Un transfert équivoque est une primitive cryptographique dans laquelle Alice a deux bits a_0 et a_1 et Bob a un bit b. Le but est que Bob apprenne a_b sans rien apprendre de a_{1-b} et sans qu'Alice n'apprenne b. Nous utilisons la définition de Shimon Even, Oded Goldreich et Abraham Lempel [15], qui est équivalente à une définition antérieure par Michael Rabin [36], tel que démontré par Claude Crépeau dans [11].

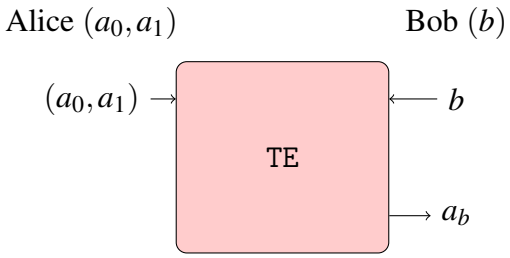


Figure 5.6 – Transfert équivoque

Lemme 6. *Un canal de communication standard équipé d'un procédé de mise en gage est au moins aussi puissant qu'un canal avec délai.*

Démonstration. Il est possible de simuler un délai en utilisant des mises en gage. Soit $\mathscr{P}$ un protocole de communication sur un canal avec délai sur entrée (x,y) où x_i est le bit envoyé à l'instant i par Alice et où y_i est le bit envoyé par Bob à l'instant i. Ces bits sont définis respectivement par la fonction $\alpha_i(x,(y_1,\ldots,y_{i-d}))$ et la fonction $\beta_i(y,(x_1,\ldots,x_{i-d}))$.

Protocole 18 : Simulation d'un protocole avec délai sur un canal de communication standard muni d'un procédé de mise en gage

1 Alice s'engage aux bits x_1 à x_d;
2 Bob s'engage aux bits y_1 à y_d;
3 $i = 1$;
4 **tant que** *Le protocole* $\mathscr{P}$ *n'est pas terminé* **faire**
5 Alice révèle le bit x_i;
6 Bob révèle le bit y_i;
7 Alice calcule $x_{i+d} = \alpha_{i+d}(x,(y_1,\ldots,y_i))$ et s'y engage;
8 Bob calcule $y_{i+d} = \beta_{i+d}(y,(x_1,\ldots,x_i))$ et s'y engage;
9 Incrémenter i;
10 **fin**
11 Alice retourne $A_i(x,(y_1,\ldots,y_{i-d}))$ et Bob retourne $B_i(y,(x_1,\ldots,x_{i-d}))$;

□

Corollaire 3. *Il est peu probable qu'un canal de communication avec délai puisse implémenter un procédé de transfert équivoque sans l'aide d'autres hypothèses.*

Démonstration. Un modèle de communication avec procédé de mise en gage est au moins aussi puissant que le modèle de communication avec délai. Or, Russel Impagliazzo et Steven Rudich [20] ont démontré que si un procédé de mise en gage était suffisant pour construire un procédé de transfert équivoque, on obtiendrait une preuve d'inégalité de P et NP. □

5.4 Avenues de recherche en cryptographie

Nishanth Chandran, Vipul Goyal, Ryan Moriarty et Rafail Ostrovsky ont étudié dans [9] l'idée d'utiliser la relativité afin d'implémenter un schème de positionnement sécuritaire. Leur première conclusion est que la relativité est insuffisante par elle même pour

y arriver, mais qu'à l'aide du modèle de récupération bornée de données, le positionnement sécuritaire est possible et peut servir à générer un échange de clés sécuritaire à l'aide de la primitive d'échange de secret à l'épreuve des intrusions bâtie par Stefan Dziemboski et Pietrzak Krzysztof [13].

Il est possible que le modèle de communication sur un canal avec délai puisse implémenter le même genre de positionnement sécuritaire, cette fois-ci en utilisant la condition de mémoire bornée plutôt que celle de récupération bornée ce qui est à notre avis plus réaliste.

CONCLUSION

Nous avons introduit un nouveau modèle de complexité de la communication dans lequel l'information est retardée de d unités de temps avant d'atteindre la fin du canal. Ce modèle est, à notre avis, plus réaliste que le modèle de la communication standard pour modéliser les communications à distance ou à haute fréquence.

Nous avons comparé la complexité de la communication avec délai de fonctions et avons conclu qu'elle ne peut différer de sa complexité de la communication standard que par un facteur multiplicatif $d/\lg d$.

Nous avons illustré un exemple classique et un exemple quantique pour lesquels une stratégie astucieuse dépendant du délai confère un avantage sur une implémentation directe d'un protocole optimal sur un canal délai.

Nous avons finalement placé le canal avec délai, d'un point de vue cryptographique, entre la primitive d'échange de bit et la primitive de mise en gage.

Avenues de recherche

Ce nouveau modèle offre de nombreuses possibilités de recherche. Il serait intéressant de déterminer la complexité de communication exacte sur un canal avec délai de fonctions utilisés dans les transactions à haute fréquence et les communications sur de longues distances.

Ensuite, il reste à montrer si un meilleur gain que le facteur $d/\lg d$ peut être fait de manière générale sur l'implémentation naïve d'un protocole de communication optimal.

Nous voudrions aussi montrer si un protocole classique optimal avec nombre de messages minimal offre toujours la stratégie optimale sur un délai ou si, comme dans le monde quantique, nous pouvons bâtir un contrexemple.

Finalement, nous avons complètement ignoré la possibilité d'inclure plus de deux participants.

BIBLIOGRAPHIE

[1] Scott Aaronson et Andris Ambainis. Quantum search of spatial regions. Dans *Proceedings of the 44th Annual IEEE Foundations of Computer Science, 2003*, pages 200–209, 2003.

[2] Harold Abelson. Lower bounds on information transfer in distributed computations. *J. ACM*, 27(2):384–392, avril 1980. ISSN 0004-5411. URL `http://doi.acm.org/10.1145/322186.322200`.

[3] Manuel Blum. Coin flipping by telephone : A protocol for solving impossible problems. *Advances in Cryptology-A Report on CRYPTO'81*, pages 23–27, 1982.

[4] Michel Boyer, Gilles Brassard, Peter Høyer et Alain Tapp. Tight bounds on quantum searching. *Fortschr. Phys*, 46:493–506, 1998.

[5] Nenad Bozinovic, Yang Yue, Yongxiong Ren, Moshe Tur, Poul Kristensen, Hao Huang, Alan E. Willner et Siddharth Ramachandran. Terabit-scale orbital angular momentum mode division multiplexing in fibers. *Science*, 340(6140):1545–1548, 2013. URL `http://www.sciencemag.org/content/340/6140/1545.abstract`.

[6] Gilles Brassard. Quantum communication complexity. *Foundations of Physics*, 33 (11):1593–1616, 2003.

[7] Gilles Brassard, David Chaum et Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37:156–189, 1988.

[8] Harry Buhrman, Richard Cleve et Avi Wigderson. Quantum vs. classical communication and computation. Dans *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 63–68, 1998.

[9] Nishanth Chandran, Vipul Goyal, Ryan Moriarty et Rafail Ostrovsky. Position based cryptography. Dans *Advances in Cryptology - CRYPTO 2009*, volume 5677 de *Lectures Notes in Computer Science*, pages 381–407. Springler Berlin / Heidelberg, shai halevi édition, 2009. ISBN 978-3-642-03355-1. URL http://dx.doi.org/10.1007/978-3-642-03356-8_23.

[10] Richard Cleve et Harry Buhrman. Substituting quantum entanglement for communication. *Physical Review A*, 56(2):1201, 1997.

[11] Claude Crépeau. Equivalence between two flavours of oblivious transfers. Dans *Advances in Cryptology—CRYPTO'87*, pages 350–354. Springer, 2006.

[12] Martin Dietzfelbinger, Juraj Hromkovič et Georg Schnitger. A comparison of two lower-bound methods for communication complexity. *Theoretical Computer Science*, 168(1):39–51, 1996. ISSN 0304-3975. URL http://www.sciencedirect.com/science/article/pii/S030439759600062X.

[13] Stefan Dziembowski et Pietrzak Krzysztof. Intrusion-resilient secret sharing. Dans *Proceedings of the 48th Annual IEEE Foundations of Computer Science*, pages 227–237, 2007.

[14] Albert Einstein. Zur elektrodynamik bewegter körper. *Annalen der physik*, 322 (10):891–921, 1905.

[15] Shimon Even, Oded Goldreich et Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, juin 1985. ISSN 0001-0782. URL http://doi.acm.org/10.1145/3812.3818.

[16] Tomas Feder, Eyal Kushilevitz, Moni Naor et Noam Nisan. Amortized communication complexity. *SIAM Journal on Computing*, 24(4):736–750, 1995. URL http://epubs.siam.org/doi/abs/10.1137/S0097539792235864.

[17] Lov K. Grover. A fast quantum mechanical algorithm for database search. Dans *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[18] Lov K. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Physical review letters*, 79(2):325, 1997.

[19] Peter Høyer et Ronald de Wolf. Improved quantum communication complexity bounds for disjointness and equality. Dans *Symposium on Theoretical Aspects of Computer Science, 2002*, pages 299–310. Springer, 2002.

[20] Russell Impagliazzo et Steven Rudich. Limits on the provable consequences of one-way permutations. Dans Shafi Goldwasser, éditeur, *Advances in Cryptology — CRYPTO' 88*, volume 403 de *Lecture Notes in Computer Science*, pages 8–26. Springer Berlin / Heidelberg, 1990. ISBN 978-0-387-97196-4. URL `http://dx.doi.org/10.1007/0-387-34799-2_2`. 10.1007/0-387-34799-2_2.

[21] Rahul Jain, Jaikumar Radhakrishnan et Pranab Sen. A lower bound for the bounded round quantum communication complexity of set disjointness. Dans *Proceedings-Annual Symposium on Foundations of Computer Science*, pages 220–229, 2003.

[22] Bala Kalyanasundaram et Georg Schintger. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5(4):545–557, 1992.

[23] Mauricio Karchmer, Ran Raz et Avi Wigderson. Super-logarithmic depth lower bounds via the direct sum in communication complexity. Dans *Proceedings of 6th Structures in Complexity Theory*, pages 299–304, 1991.

[24] Mauricio Karchmer et Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM Journal on Discrete Mathematics*, 3(2):255–265, 1990.

[25] Adrian Kent. Coin tossing is strictly weaker than bit commitment. *Physical Review Letters*, 83(25):5382, 1999.

[26] Adrian Kent. Unconditionally secure bit commitment. *Physical Review Letters*, 83:1448–1450, 1999.

[27] Adrian Kent. Secure classical bit commitment using fixed capacity communication channels. *Journal of Cryptology*, 18:313–335, 2005.

[28] Hartmut Klauck. Quantum communication complexity. *arXiv preprint quant-ph/0005032*, 2000.

[29] Hartmut Klauck, Ashwin Nayak, Amnon Ta-Shma et David Zuckerman. Interaction in quantum communication and the complexity of set disjointness. Dans *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 124–133, 2001.

[30] Eyal Kushilevitz et Noam Nissan. *Communication Complexity*. Cambridge University Press, 1997.

[31] Thomas Lengauer. Handbook of theoretical computer science. pages 835–866. MIT Press, Cambridge, MA, USA, 1990. ISBN 0-444-88071-2. URL `http://dl.acm.org/citation.cfm?id=114872.114888`.

[32] NASA/JPL. Mars science laboratory : Mission : Rover : Eyes and other senses : Four engineering hazcams (hazard avoidance cameras), 2009. URL `http://mars.jpl.nasa.gov/msl/mission/rover/eyesandother/`.

[33] Michael A. Nielsen et Isaac L. Chuang. *Quantum Computation and Quantum Information (Cambridge Series on Information and the Natural Sciences)*. Cambridge University Press, première édition, 2004.

[34] Noam Nisan et Avi Widgerson. Rounds in communication complexity revisited. Dans *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing*, STOC '91, pages 419–429, New York, NY, USA, 1991. ACM. ISBN 0-89791-397-3. URL http://doi.acm.org/10.1145/103418.103463.

[35] Christos H. Papadimitriou et Michael Sipser. Communication complexity. Dans *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, STOC '82, pages 196–200, San Francisco, California, United States, 1982. ACM. ISBN 0-89791-070-2. URL http://doi.acm.org/10.1145/800070.802192.

[36] Michael O. Rabin. How to exchange secrets with oblivious transfer. *Technical Report TR-81, Aiken Computation Lab, Harvard University*, 1981.

[37] Ran Raz. Exponential separation of quantum and classical communication complexity. Dans *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 358–367, 1999.

[38] Alexander A. Razborov. Quantum communication complexity of symmetric predicates. *Izvestiya : Mathematics*, 67(1):145–159, 2003.

[39] Clark D. Thompson. Area-time complexity for VLSI. Dans *Proceedings of the eleventh annual ACM symposium on Theory of computing*, pages 81–88, 1979.

[40] Alexander D. Wissner-Gross et Cameron E. Freer. Relativistic statistical arbitrage. *Physical Review E*, 82(5):1–7, 2010. URL http://link.aps.org/doi/10.1103/PhysRevE.82.056104.

[41] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). Dans *Proceedings of the eleventh annual ACM symposium on Theory of computing*, STOC '79, pages 209–213, Atlanta, Georgia, Uni-

ted States, 1979. ACM. URL http://doi.acm.org/10.1145/800135.804414.