# Couplage maximum dans les modèles de streaming

Omar Fawzi

Stage encadré par Frédéric Magniez
dans l'équipe Algorithmique et Complexité du LRI

24 août 2008

## Le contexte général

Au cours des dix dernières années, le modèle de streaming a fait l'objet de nombreux travaux, motivés par le besoin de calculs dynamiques de statistiques sur des masses de données, mais aussi par des avancées théoriques comme par exemple [FM83] et [AMS99] qui a obtenu le prix Gödel. La particularité de ce modèle est une forte contrainte sur la quantité de mémoire disponible qui est sous-linéaire en la taille de l'entrée, et un petit nombre de lectures séquentielles (passes) de l'entrée. Cependant, les problèmes bien étudiés se restreignaient jusqu'à récemment majoritairement à une catégorie spécifique de problèmes : le calcul approché de statistiques. L'étude de problèmes sur les graphes dans ce modèle a commencé récemment à prendre de l'importance [BYKS02, FKM$^+$05a, FKM$^+$05b, McG05, Zel08]. Les problèmes sur les graphes sont en général difficiles dans ce modèle et ne peuvent pas être résolus en espace poly-logarithmique, la contrainte sur l'espace est en général relaxée pour autoriser une taille mémoire de l'ordre du nombre de sommets du graphes (qui est potentiellement de l'ordre de la racine carrée de la taille de l'entrée). Dans ce modèle, le problème du couplage de taille maximale a été étudié dans [FKM$^+$05a, McG05, Zel08]. Dans [FKM$^+$05a], un algorithme d'approximation du couplage maximum dans un graphe biparti est donné, et un algorithme d'approximation du couplage de taille maximale dans un graphe avec poids est présenté. Dans [McG05, Zel08], ces algorithmes sont améliorés pour donner de meilleures garanties sur le facteur d'approximation.

## Le problème étudié

Le problème général étudié ici est le problème de trouver un couplage de taille maximale dans un graphe dans les modèles de streaming. Dans le modèle simple à une passe, le meilleur algorithme connu est l'algorithme glouton qui calcule un couplage dont la taille est au pire la moitié de celle d'un couplage de taille maximale. Cependant, il n'existe aucune preuve d'optimalité de cet algorithme pour le modèle. Un problème naturel dans ce cadre est de chercher à montrer l'optimalité de cet algorithme ou de trouver un meilleur algorithme. Ce problème ne semble pas avoir été abordé dans la littérature, sans doute (rétrospectivement) à cause de sa difficulté. Mais les études concernant le problème du couplage maximum en streaming ont plutôt été poursuivies dans l'objectif d'obtenir des schémas d'approximations en plusieurs passes. Par exemple, l'algorithme de [McG05] présente un algorithme avec facteur d'approximation $1 - \epsilon$ utilisant un nombre de passes qui ne dépend que de $\epsilon$.

Les autres problèmes abordés ici sont la comparaison des différents modèles de streaming en ce qui concerne le calcul de couplage maximum. Par exemple, nous avons cherché une variante du modèle dans laquelle il est possible de calculer un couplage de taille raisonnable (par exemple un couplage maximal) avec une mémoire poly-logarithmique.

### La contribution proposée

Pour tenter de trouver un algorithme trouvant en une passe un couplage qui donne un rapport d'approximation meilleur que $\frac{1}{2}$, j'ai essayé plusieurs méthodes dont principalement deux : garder plusieurs couplages en mémoire et les combiner, ou partitionner le graphe en paquets de diverses façons. Ces méthodes n'aboutissant pas, il m'a semblé que ce modèle ne permet pas de faire beaucoup mieux qu'un couplage maximal. J'ai donc cherché une borne inférieure pour ce problème. D'abord j'ai cherché à démontrer que le problème de trouver un couplage de taille maximale est impossible à résoudre dans notre modèle. Puis j'ai essayé de généraliser cette preuve d'impossibilité au problème d'approximation. Malheureusement, cette technique ne donne que l'impossibilité d'approximation à un facteur sous-linéaire.

J'ai donc cherché à prouver une impossibilité dans un modèle restreint, ceci donna un preuve d'impossibilité d'approximation à un facteur constant donné mais avec peu de mémoire. Cette preuve donne cependant une compréhension du problème sous un autre angle et semble indiquer la possibilité de l'existence d'un algorithme. La notion de paire $\epsilon$-régulière fait alors son apparition, comme un cas spécial où le problème devient plus facile.

Dans les modèles de streaming plus compliqués, notamment dans un modèle introduit récemment avec une primitive de tri, les algorithmes parallèles peuvent être traduits dans ce modèle et donner des algorithmes performants n'utilisant qu'une mémoire poly-logarithmique.

### Les arguments en faveur de sa validité

Les bornes inférieures proposées dans ce rapport et les bornes inférieures en général pour les modèles de streaming sont des bornes inférieures sur la *communication*, c'est-à-dire qu'avec une petite mémoire, il est impossible de transmettre une certaine information du début du stream à la fin. Ce type de borne inférieure ne parait pas suffire pour résoudre complètement notre problème. La difficulté peut aussi venir du fait que l'ordre d'arrivée du stream est arbitraire, ou encore de la difficulté du calcul lui même avec peu de ressources.

### Le bilan et les perspectives

Ce travail a commencé une étude approfondie du problème du couplage maximum dans le modèle simple de streaming à une passe. Quelques bornes inférieures ont été montrées mais celles-ci restent loin du meilleur algorithme connu. Quelques résultats laissent croire que l'existence d'algorithmes d'approximation avec des facteurs d'approximation meilleures que $\frac{1}{2}$ est possible. La piste du lemme de régularité de Szemerédi semble prometteuse. D'autre part, de meilleures bornes inférieures dans les modèles à plusieurs passes seraient très intéressantes.

# 1 Préliminaires

## 1.1 Le modèle

Les modèles de streaming sont définis dans le but de modéliser le fait que l'entrée arrive très rapidement et que la mémoire pour stocker l'entrée est petite. Les algorithmes doivent alors résoudre le problème en un petit nombre de passes avec un espace mémoire sous-linéaire en la taille de l'entrée. L'origine de ces modèles est la multiplication des "data streams" ou flux de données, par exemple pour des données concernant le trafic internet, des données financières, mais aussi des données en provenance de réseaux de capteurs. Souvent, ce que l'on souhaite calculer sur un stream est de type statistique : calculer des agrégats, les éléments qui apparaissent le plus souvent, détecter des anomalies etc... Les solutions de ce type de problème sont le plus souvent *approchées* et *probabilistes*.

Au fur et à mesure que le modèle a pris de l'importance d'autres problèmes ont commencé à être étudiés dans ce modèle, en particulier les problèmes de graphes. De grands graphes apparaissent naturellement sous la forme de stream, par exemple un *call-graph* est un graphe dont les sommets représentent les numéros de téléphone et les arêtes représentent des communications téléphoniques pendant un intervalle de temps donné. Un stream pour un graphe est une liste de ses arêtes dans un ordre arbitraire. Un graphe sera noté en général $G = (V, E)$ où $V$ l'ensemble de ses sommets et $E$ l'ensemble des ses arêtes, avec $m = |E|$. On supposera que le nombre de sommets du graphes $|V|$ est donné au début du stream, par contre le longueur du stream $m$ est inconnue avant le passage du stream.

**Definition 1.** Un stream représentant un graphe $G = (V, E)$ est une suite d'arêtes de $E$ dans un ordre arbitraire.

Les deux contraintes imposées par le modèle de streaming sont le petit nombre de passes sur l'entrée et le peu de mémoire, on peut voir ce modèle comme un intermédiaire entre les algorithmes dynamiques qui doivent maintenir une propriété sur un objet dynamique mais n'ont pas de contrainte sur l'espace mémoire, et les algorithmes en espace poly-logarithmique mais sans contrainte sur le nombre de passes sur l'entrée. De nombreux algorithmes dynamiques sur les graphes sont connus, mais à l'inverse, les problèmes de graphes sont considérés comme difficiles en espace poly-logarithmique. Le modèle peut être défini comme suit

**Definition 2.** Un *algorithme de streaming* en $p$ passes avec mémoire $s$ est un algorithme qui fait $p$ passes séquentielles sur l'entrée en utilisant une mémoire de travail de taille $s$, et écrit sa sortie sur la mémoire de travail.

Généralement, les algorithmes de streaming utilisent un espace poly-logarithmique et un nombre de passes constant (de préférence une seule passe). Cependant, sur les graphes, un espace poly-logarithmique est souvent trop faible. Par exemple, ne serait-ce que pour décider s'il existe un chemin de longueur 2 entre deux sommets $x$ et $y$, une mémoire $\Omega(|V|)$ (c'est-à-dire de l'ordre du nombre de sommets) est nécessaire [KS92].

Mais la taille de l'entrée est de l'ordre de $m$ qui est potentiellement quadratique en $|V|$ et donc un algorithme qui n'utiliserait qu'un espace $O(|V| \cdot \text{polylog}(|V|))$ serait intéressant. Ceci nous amène à considérer le modèle semi-streaming introduit dans [Mut06] : c'est le modèle de streaming pour les problèmes de graphes avec un espace $s = O(|V| \cdot \text{polylog}(|V|))$, qui est sous-linéaire pour les graphes denses par exemple. Cette mémoire nous permet de garder un représentant compact du graphe, les problèmes qui seront résolubles dans ce modèle seront

donc les propriétés sur les graphes qui peuvent bien être résumées. Dans le modèle semi-streaming des algorithmes sont connus pour le calcul approché de plus court chemin, le calcul d'un arbre couvrant de poids minimum et le calcul des composantes connexes [FKM+05a] par exemple. Beaucoup de ces algorithmes s'inspirent de travaux comme [EGIN97] où la notion de *sparsification* est introduite pour obtenir des algorithmes dynamiques efficaces, il s'agit d'une représentation par un graphe peu dense du graphe considéré qui conserve certaines propriétés.

Le problème du couplage maximum, pour lequel les techniques de sparsification ne semblent pas s'appliquer, est aussi étudié dans [FKM+05b] qui donne un algorithme d'approximation pour les graphes bipartis, [McG05] donne des algorithmes d'approximation pour les graphes sans et avec poids. [Zel08] donne un meilleur algorithme d'approximation en une passe pour les graphes avec poids.

## 1.2 Complexité de communication

Pour prouver des résultats d'impossibilité dans le modèle de streaming, une méthode est d'utiliser la complexité de communication. En général ce qui est difficile avec une petite mémoire, c'est de faire passer de l'information d'une partie de l'entrée à l'autre. Ainsi, les preuves de bornes inférieures dans les modèles de streaming passent par des réductions à un problème de communication, dont on calcule la complexité de communication.

La complexité de communication a été introduite par Yao dans [Yao79] qui étudiait le problème du calcul de la valeur d'une fonction sur une entrée qui est partagée entre Alice et Bob. Pour $f : X \times Y \to Z$, Alice choisit $x \in X$, Bob $y \in Y$ et le but est de calculer $f(x, y)$ avec le minimum de communication entre Alice et Bob. Le nombre de bits utilisés par un protocole est le nombre de bits de communication utilisés sur la pire entrée. La complexité de communication $D(f)$ d'un fonction $f$ est le minimum sur l'ensemble des protocoles déterministes calculant $f$ du nombre de bits utilisés par ce protocole. La complexité de communication probabiliste $R_\delta(f)$ est la communication minimum d'un protocole qui calcule $f$ en se trompant avec probabilité $\leq \delta$.

La communication peut être à sens unique, c'est-à-dire que ce n'est qu'Alice qui envoie des données à Bob et c'est Bob qui doit calculer le résultat de la fonction, ou à double sens c'est-à-dire qu'Alice et Bob peuvent tous les deux s'échanger des informations.

Le seul résultat de complexité de communication utile dans ce rapport est un résultat sur la fonction Disj : $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}$

$$\textsc{Disj}(x, y) = \begin{cases} 0 & \text{si il existe } i \in \{1, \ldots, n\} \text{ tel que } x_i = y_i = 1 \\ 1 & \text{sinon} \end{cases}$$

**Théorème 1** ([KS92]). *La complexité de communication probabiliste de* Disj *est* $\Omega(n)$.

## 1.3 Le problème du couplage maximum

**Definition 3.** Un couplage d'un graphe $G = (V, E)$ est un sous-ensemble $M \subset E$ tel que aucun couple d'arêtes de $M$ ne partage un sommet.

Le problème que l'on se pose est de trouver un couplage de grande taille. Remarquons qu'un algorithme glouton qui rajoute les arêtes si possible ne donne pas nécessairement un couplage de taille maximum. En effet, comme le montre la figure 1, le couplage $M$ est maximal (aucune arête ne peut lui être rajoutée) et est de taille 1, alors qu'il existe un couplage de taille 2.

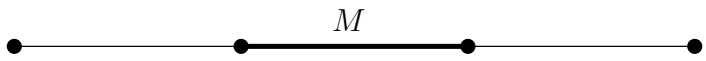


**Fig. 1** – *Un couplage maximal n'est pas nécessairement de taille maximale*

On notera dans la suite $\textsc{MaxMat}(G)$ la taille d'un couplage de taille maximale (ou couplage maximum) de $G$. On dit qu'un graphe a un couplage parfait si tous ses sommets sont couplés, i.e. $\textsc{MaxMat}(G) = \frac{|V|}{2}$

Remarquons qu'un couplage maximal ne peut pas être arbitrairement mauvais. En effet,

**Propriété 1.** *Soit $G$ un graphe pour lequel $M$ est un couplage maximal, alors $|M| \geq \frac{1}{2}\textsc{MaxMat}(G)$.*

*Démonstration.* Soit $M'$ un couplage maximum. Considérons une arête $(u, v) \in M'$, alors au moins un des deux sommets $u$ et $v$ est couvert par $M$. En effet, si ni $u$ ni $v$ ne sont couverts alors l'existence de l'arête $(u, v)$ dans le graphe $G$ contredit le caractère maximal de $M$. Donc $M$ couvre au moins autant de sommets que $M'$ a d'arêtes. On a alors $|M| \geq \frac{1}{2}|M'|$. □

L'étude de ce problème a fait l'objet de très nombreux travaux depuis les années 50 qui sont exposés dans [LP86]. Plusieurs variantes du problème existent, par exemple, pour des graphes dont les arêtes ont des poids, l'objectif est alors de trouver un couplage de poids total maximal.

Le problème de trouver un couplage maximum dans un graphe se résout en temps polynomial, cependant, les algorithmes existants ne sont pas triviaux et ils ont mis du temps à se mettre en place.

Une notion importante concernant les couplages est celle de chemin augmentant. Il s'agit pour un couplage, d'un chemin alternant entre des arêtes du couplage et des arêtes qui n'y sont pas.

**Definition 4.** Soit $G$ un graphe et $M$ un couplage de $G$. On dit qu'un chemin $C = v_1, \ldots, v_m$ est *alternant* par rapport à $M$ si pour tout $1 \leq i \leq m-2$,

$$(v_i, v_{i+1}) \in M \text{ si et seulement si } (v_{i+1}, v_{i+2}) \notin M.$$

Un *chemin augmentant* pour $M$ est un chemin alternant tel que le premier et le dernier sommets ne sont pas couverts par $M$.

Un chemin augmentant $C$ pour un couplage $M$ donne un autre couplage $M'$ en supprimant les arêtes de $M \cap C$ et en rajoutant les arêtes de $C - M$. Il est donc clair que si un couplage $M$ admet un chemin augmentant alors, il n'est pas de taille maximum. Il se trouve que la réciproque est aussi vraie et c'est ce qui permet de donner des algorithmes pour résoudre le problème, en augmentant un couplage de départ jusqu'à ce que ce ne soit plus possible.

**Théorème 2** (Berge)**.** *Soit $M$ un couplage du graphe $G$. Alors $M$ est un couplage maximum si et seulement si $G$ n'a pas de chemin augmentant pour $M$.*

De cette propriété, on peut déduire une classe d'algorithmes qui construisent un couplage de manière incrémentale en commençant avec un couplage simple (par exemple un couplage maximal), puis en augmentant la taille du couplage tant que c'est possible en trouvant un

chemin augmentant. Cependant chercher tous les chemins augmentants est une opération délicate, qui est plus simple pour les graphes bipartis.

Le meilleur algorithme connu pour trouver un couplage maximum dans un graphe général $G = (V, E)$, a un coût de $O(\sqrt{|V|}|E|)$ dans [MV80]. Pour un graphe avec poids, l'algorithme le plus rapide pour calculer un couplage de poids maximal décrit dans [Gab90] a un coût de $O(|V||E| + |V|^2 \log |V|)$ .

Dans la suite, on va considérer des graphes bipartis dont l'ensemble des sommets sera noté $(L, R)$ avec $|L| = |R| = n$ pour la suite sauf indication contraire. La convention générale de notation pour $n$ dans la suite est $n = \Theta(|V|)$, ainsi parfois on notera $|V| = 4n - 2$ ou $|V| = 4n$ pour alléger les écritures. La *deficiency* d'un graphe biparti est une notion utile qu'on utilisera par la suite.

**Definition 5.** Soit $G = (L, R, E)$ un graphe biparti. Pour $X \subset L$, on note $def_G(X) = |X| - |\Gamma(X)|$ où $\Gamma(X)$ représente le voisinage de $X$ dans $G$. Pour le graphe $G$,

$$def(G) = \max_{X \subset L} def_G(X).$$

L'intérêt de cette notion est qu'elle caractérise d'une manière assez simple la taille du couplage maximum d'un graphe biparti.

**Théorème 3** (Hall)**.** *La taille d'un couplage maximum d'un graphe biparti $G = (L, R, E)$ est $|L| - def(G)$.*

Il existe une multitude de formulations (parfois appelées théorème de König ou théorème des mariages) et de preuves de ce résultat.

Pour finir, introduisons un lemme qui va nous être utile par la suite

**Lemme 1.** *Soit $G = (V, E)$ un graphe, et $u$ un sommet de degré 1 avec une arête $(u, v)$. Alors il existe un couplage maximum qui contient $(u, v)$.*

*Démonstration.* Soit $M$ un couplage maximum quelconque. Si $(u, v) \in M$, alors on a fini. Si $(u, v) \notin M$, alors $u$ n'est pas couplé dans $M$ puisque $u$ est de degré 1, et $v$ est couplé à un certain $x$. On construit alors $M \cup (u, v) - (v, x)$ qui est bien un couplage et qui a la même taille que $M$. □

## 2 Couplage maximum dans le modèle de streaming à une passe

En une passe, l'algorithme le plus simple est l'algorithme glouton qui rajoute une arête au couplage si c'est possible.

**Fait 1.** *L'algorithme glouton* GreedyMat *est un algorithme de streaming en 1 passe avec mémoire $O(|V| \log |V|)$ qui donne un couplage tel que pour tout graphe $G$, $|\textsc{GreedyMat}(G)| \geq \frac{1}{2}\textsc{MaxMat}(G)$.*

Une question naturelle ici est de voir si il est possible de faire mieux que cet algorithme.

## 2.1 Complexité de communication

### 2.1.1 Borne inférieure pour le calcul exact

Pour commencer, essayons de montrer qu'un algorithme en une passe qui calcule un couplage de taille maximum doit utiliser une mémoire $\Omega(|V|^2)$. Pour ceci, on montre une borne inférieure sur la communication nécessaire : pour résoudre le problème, presque toutes les informations du stream doivent être maintenues.

Pour formaliser le problème, on le définit en terme de problème de communication. Alice et Bob détiennent chacun une partie d'un graphe biparti, Alice doit envoyer un message à Bob et Bob doit décider si le graphe a un couplage parfait ou pas. La répartition des arêtes est fixée comme suit. Le graphe a $4n-2$ sommets, $2n-1$ à gauche et $2n-1$ à droite. Alice détient les arêtes entre les $n$ premiers sommets de gauche et les $n$ premiers sommets de droite (on appellera ces sommets les sommets d'Alice), et Bob détient les autres arêtes. Cette répartition est illustrée dans la figure 2. On cherche alors la complexité de communication à sens unique de la fonction

$$\textsc{PerfectMat}_n(G,H) = \begin{cases} 1 & \text{si le graphe formé des arêtes } G \cup H \text{ a un couplage parfait} \\ 0 & \text{sinon} \end{cases}$$

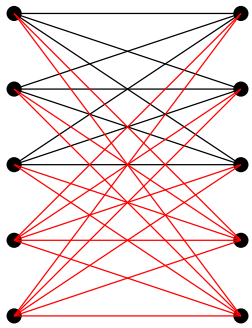

**Fig. 2** – *La répartition des arêtes entre Alice et Bob, l'entrée d'Alice $G$ correspond à un sous-ensemble des arêtes noires et l'entrée de Bob $H$ correspond à un sous-ensemble des arêtes rouges*

Si un algorithme en une passe décide si le graphe a un couplage parfait, alors il fournit un protocole pour calculer la fonction $\textsc{PerfectMat}_n$ : Alice fait tourner l'algorithme sur son graphe, et envoie l'état de la mémoire à Bob qui continue l'exécution de l'algorithme en utilisant la mémoire qu'Alice lui a envoyée. On en déduit donc que l'espace utilisé par un algorithme en une passe est supérieur à la complexité de communication de la fonction $\textsc{PerfectMat}_n$.

**Lemme 2.** *Soit $G_1$ et $G_2$ deux graphes bipartis distincts d'Alice. Alors il existe un graphe $H$ de Bob tel que le graphe $G_1 \cup H$ a un couplage parfait et $G_2 \cup H$ n'en a pas (ou l'inverse), i.e.* $\textsc{PerfectMat}_n(G_1,H) \neq \textsc{PerfectMat}_n(G_2,H)$.

*Démonstration.* Si $G_1$ et $G_2$ sont distincts, alors il existe une arête qui est uniquement dans un des deux graphes. On peut supposer que $(u,v)$ est dans $G_1$ mais pas dans $G_2$. Construisons alors le graphe $H$ de Bob. $H$ est composé d'un couplage parfait entre les sommets $\{1,\dots n\} - \{u\}$ de gauche et les sommets $\{n+1,\dots 2n-1\}$ de droite et un couplage parfait entre $\{n+$

$1, \dots 2n-1\}$ de gauche et les sommets $\{1, \dots n\} - \{v\}$ de droite. Cette construction est illustrée dans la figure 3.

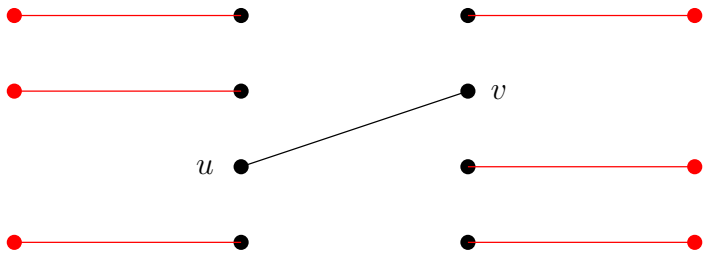


**Fig. 3** – *Deux graphes qui diffèrent en une arête peuvent être différenciés par un graphe $H$, représenté ici en rouge. Pour plus de clarté, les sommets de Bob sont dessinés de part et d'autre des sommets d'Alice*

Montrons d'abord que $G_2 \cup H$ n'a pas de couplage parfait. Les sommets $\{n+1, \dots 2n-1\}$ de gauche et de droite sont tous de degré au plus 1, donc par le lemme 1 il existe un couplage de taille maximale qui contient toutes les arêtes de $H$. Les arêtes de $H$ couplent tous les sommets du graphe sauf $u$ et $v$. Puisque l'arête $(u, v)$ est la seule à pourvoir couplé $u$ et $v$ ensemble, et que $(u, v) \notin G_2$ et donc $(u, v) \notin G_2 \cup H$, les arêtes de $H$ forment un couplage de taille maximale de $G_2 \cup H$. La taille maximale d'un couplage de $G_2 \cup H$ est donc $2n-2$, et le nombre de sommets du graphe est $4n-2$. En conclusion, $G_2 \cup H$ n'a pas de couplage parfait.

A l'inverse, pour $G_1 \cup H$, rajouter $(u, v)$ aux arêtes de $H$ forme un couplage parfait du graphe. □

**Remarque** : cette construction sera utilisée plusieurs fois dans ce rapport. Lorsqu'on parlera de la construction du graphe $H$ du lemme 2 relatif à $G_1$ et $G_2$, il s'agira de prendre une arête $(u, v)$ qui est dans $G_1$ et pas dans $G_2$ (ou l'inverse) et de construire $H$ comme indiqué dans la figure 3.

On en déduit immédiatement

**Lemme 3.** *La complexité déterministe de communication à sens unique de la fonction* PerfectMat$_n$ *est au moins* $n^2$.

*Démonstration.* D'après le lemme 2, pour deux entrées distinctes d'Alice $G_1$ et $G_2$ le message envoyé par Alice doit être différent. Le nombre de graphes d'Alice est le nombre de graphe bipartis avec $n$ sommets à gauche et $n$ à droite, il y en a donc $2^{n^2}$. Alice doit donc pouvoir envoyer $2^{n^2}$ messages distincts. Le nombre de bits du message envoyé doit donc être $\log 2^{n^2} = n^2$. □

Un algorithme de streaming en une passe utilisant pour un graphe $G = (V, E)$ $s(|V|)$ bits de mémoire se traduit en un protocole pour le problème PerfectMat$_n$ utilisant $s(4n-2)$ bits de communication, puisque la taille du graphe du problème PerfectMat$_n$ est $4n-2$. Le lemme précédent montre alors que pour tout $n$, $s(4n-2) \geq n^2$.

**Corollaire 1.** *Un algorithme décidant de l'existence d'un couplage parfait en une passe des arêtes du graphe utilise une mémoire de taille* $\Omega(|V|^2)$.

### 2.1.2 Borne inférieure pour le calcul approché

On cherche maintenant à améliorer cette borne inférieure pour dire que même trouver un couplage de taille proche de la taille maximum est difficile. On va utiliser la construction de la section précédente et la répliquer pour construire un grand nombre de graphes d'Alice, tel que pour toute paire de graphes $G_1, G_2$, il existe un graphe de Bob qui fait que l'un des deux graphes $G_1 \cup H$ et $G_2 \cup H$ a un couplage maximum beaucoup plus grand que l'autre.

Le graphe considéré est un graphe biparti $(L, R)$ avec $L = L_1 \cup L_2$ et $R = R_1 \cup R_2$ avec $|L_1| = |L_2| = |R_1| = |R_2| = n$. Le nombre de sommets du graphe est alors $4n$. On va également considérer un problème de communication entre Alice et Bob. La répartition des arêtes entre les deux parties sera similaire à celle de la section précédente : Alice contrôle les arêtes de $L_1 \times R_1$, et Bob contrôle toutes les autres arêtes.

On définit donc un problème de communication approché, on veut maintenant calculer un grand couplage du graphe et non seulement décider l'existence d'un couplage parfait.

$\textsc{AppMaxMat}_n^k(G, H)$ :

- Entrée : Un sous-ensemble d'arêtes $G$ (graphe d'Alice) de $L_1 \times R_1$ et un sous-ensemble d'arêtes $H$ (graphe de Bob) de $L \times R - L_1 \times R_1$.
- Sortie : Bob calcule un couplage $M$ de $G \cup H$ tel que $|M| \geq \textsc{MaxMat}(G \cup H) - k$.

Remarquons qu'ici, nous ne cherchons pas à montrer une borne inférieure sur une fonction particulière, mais sur un problème pour lequel une entrée peut avoir plusieurs sorties possibles.

La stratégie pour démontrer une borne inférieure sur la communication est de trouver un grand ensemble de graphes d'Alice, tel que pour toute paire de graphes $G_1, G_2$, il existe un graphe $H$ de Bob tel que pour tout protocole $P$ qui résout $\textsc{AppMaxMat}_n^k$, les sorties du protocole sur les entrées $(G_1, H)$ et $(G_2, H)$ doivent être différentes. Pour garantir que deux graphes ont des sorties différentes, on assure que

$$|\textsc{MaxMat}(G_1 \cup H) - \textsc{MaxMat}(G_2 \cup H)| > k. \tag{1}$$

Ainsi, les messages qu'Alice envoie pour $G_1$ et pour $G_2$ doivent être différents, ceci quelque soit le protocole utilisé. En effet, si les messages pour $G_1$ et $G_2$ étaient les mêmes, les sorties d'un protocole sur $(G_1, H)$ et $(G_2, H)$ serait les mêmes. Les graphes $G_1 \cup H$ et $G_2 \cup H$ aurait donc un couplage commun $M$ et celui-ci vérifie $|M| \geq \textsc{MaxMat}(G_1 \cup H) - k$ et $|M| \geq \textsc{MaxMat}(G_2 \cup H) - k$. Mais puisque $|M| \leq \textsc{MaxMat}(G_1 \cup H)$ et $|M| \leq \textsc{MaxMat}(G_2 \cup H)$, on obtient $\textsc{MaxMat}(G_1 \cup H) \leq \textsc{MaxMat}(G_2 \cup H) + k$ et $\textsc{MaxMat}(G_2 \cup H) \leq \textsc{MaxMat}(G_1 \cup H) + k$, ce qui contredit la propriété 1. On cherche donc une grande famille de graphes telle que chaque paire de graphes $G_1, G_2$ vérifie la propriété 1.

Commençons par donner une construction simple qui donne une borne inférieure pour la communication de $\Omega((\frac{n}{k})^2)$ pour $\textsc{AppMaxMat}_n^k$.

Le grand ensemble de graphes d'Alice est défini comme suit. Un graphe sera la juxtaposition de $k+1$ copies d'un graphe biparti ayant $\frac{n}{k+1}$ sommets de gauche comme de droite. Cette construction est illustrée dans la figure 4.

Le choix de répéter $k+1$ fois le même graphe sert à garantir que lorsqu'on compare deux graphes différents, ils sont différents dans les $k+1$ composantes du graphes.

$$\mathcal{G} = \{k+1 \text{ copies de } g | g \text{ biparti avec } 2\frac{n}{k+1} \text{ sommets}\}$$

Pour $G_1 \in \mathcal{G}$ et $G_2 \in \mathcal{G}$, $H$ est construit de la manière suivante : vu que $G_1 \neq G_2$, les motifs de construction $g_1$ et $g_2$ de $G_1$ et $G_2$ sont différents, on construit alors le graphe de

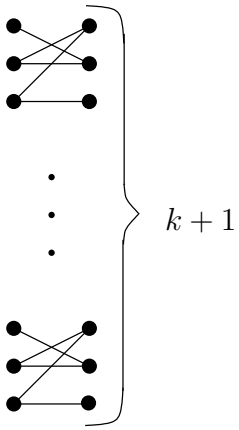


**Fig. 4** – *Construction de $\mathcal{G}$ : répétition d'un graphe de taille $\frac{n}{k+1}$*

Bob $h$ relatif à $g_1$ et $g_2$ comme décrit dans la preuve du lemme 2, et ceci est fait pour chaque composante du graphe, comme indiqué dans la figure 5.

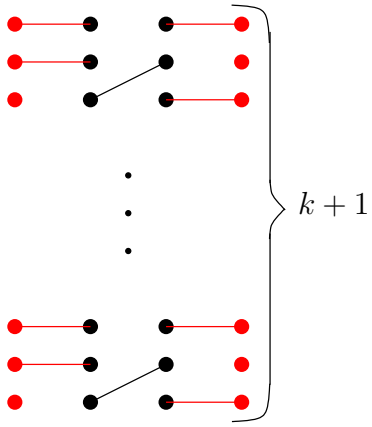


**Fig. 5** – *Construction de $H$ pour $G_1$ et $G_2$*

Montrons alors que $\textsc{MaxMat}(G_1 \cup H) \geq \textsc{MaxMat}(G_2 \cup H) + (k+1)$ (en fait on a même égalité). En effet, on peut toujours supposer que les arêtes de $H$ sont dans un couplage maximum d'après le lemme 1, vu que les sommets de $L_2$ et de $R_2$ sont de degré au plus 1. On a donc entre $G_1 \cup H$ et $G_2 \cup H$ une différence de 1 en taille de couplage pour chaque composante du graphe et donc une différence de $k+1$ au total. D'où on déduit que $G_1$ et $G_2$ doivent envoyer des messages différents.

La taille de notre ensemble de graphe est $|\mathcal{G}| = 2^{(\frac{n}{k+1})^2}$ et donc on obtient une borne inférieure pour la communication d'un protocole qui résout $\textsc{AppMaxMat}_n^k$ de $\Omega((\frac{n}{k})^2)$.

Mais on peut obtenir une meilleure borne.

**Théorème 4.** *Soit $\alpha < \frac{1}{\sqrt{32}}$ une constante. Pour tout $n$ assez grand et $k \leq \alpha n$, un protocole pour $\textsc{AppMaxMat}_n^k$ utilise au moins $\frac{n^2}{k}(\frac{1}{16} - 2\alpha^2)$ bits de communication.*

*Démonstration.* Le principe est le même, on va construire un grand ensemble $\mathcal{G}$,

$$\forall G_1, G_2 \in \mathcal{G}, G_1 \neq G_2 \Rightarrow \exists H, |\textsc{MaxMat}(G_1 \cup H) - \textsc{MaxMat}(G_2 \cup H)| > k.$$

On va procéder de la même manière en découpant le graphe en composantes, sauf qu'on aura $4k$ composantes au lieu de $k+1$. Dans chaque composante, on voudrait mettre un graphe de sorte que pour tout couple de graphe de $\mathcal{G}$, ils diffèrent en au moins $2k+1$ composantes. Supposons cet ensemble construit, on aura donc pour tout couple de graphes $G_1, G_2 \in \mathcal{G}$, $G_1$ diffère de $G_2$ dans au moins $2k+1$ composantes, et donc, quitte à échanger $G_1$ et $G_2$, on peut supposer qu'il y a au moins $k+1$ composantes du graphe où $G_1$ à une arête que $G_2$ n'a pas. On construit donc $H$ tel que pour chacune de ces $k+1$ composantes, on fait comme dans le lemme 2 et pour les autres composantes, on rajoute un couplage parfait entre les sommets des composantes et des sommets de Bob. De cette manière $\textsc{MaxMat}(G_1 \cup H) = \textsc{MaxMat}(G_2 \cup H) + (k+1)$.

Il reste à construire l'ensemble vérifiant la propriété que pour tout couple de graphes avec $4k$ composantes, il existe au moins $2k+1$ composantes où ils diffèrent. Ce problème peut être vu comme un problème sur les mots où notre alphabet est l'ensemble des graphes possibles dans une composante (alphabet de taille $2^{(\frac{n}{4k})^2}$), on veut construire un ensemble de mots de longueur $4k$ telle que chaque paire de mots diffère en au moins $2k+1$ positions. On utilise le lemme suivant

**Lemme 4.** *Soit $\mathcal{A}$ un alphabet fini. Il existe un ensemble $E \subset \mathcal{A}^{4k}$, $|E| \geq \lfloor |\mathcal{A}|^k 2^{-2k} \rfloor$ mots tel que*

$$\forall w_1, w_2 \in E, w_1 \neq w_2 \Rightarrow d_H(w_1, w_2) > 2k$$

*où $d_H$ est la distance de Hamming, $d_H(w_1, w_2) = |\{i, w_1(i) \neq w_2(i)\}|$.*

*Démonstration.* On utilise la méthode probabiliste. On met la distribution uniforme sur l'ensemble des mots de taille $4k$. On choisit $r$ mots aléatoires $w_1, \dots w_r$

On a alors

$$\begin{aligned}
\mathbb{P}(\exists i, j \in \{1, \dots r\}, d_H(w_i, w_j) \leq 2k) &\leq \binom{r}{2} \mathbb{P}(d_H(w_1, w_2) \leq 2k) \\
&\leq \frac{1}{2} r^2 \frac{\sum_{l=0}^{2k} \binom{4k}{l} (|\mathcal{A}| - 1)^l}{|\mathcal{A}|^{4k}} \\
&\leq \frac{1}{2} r^2 \frac{2^{4k} |\mathcal{A}|^{2k}}{|\mathcal{A}|^{4k}} \\
&= \frac{1}{2} r^2 2^{4k} |\mathcal{A}|^{-2k}.
\end{aligned}$$

Donc si on prend $r = \lfloor |\mathcal{A}|^k 2^{-2k} \rfloor$, cette probabilité va être strictement inférieure à 1 et donc un ensemble $E$ vérifiant les conditions demandées existe bien. □

En appliquant ce lemme on obtient donc un ensemble $\mathcal{G}$ qui a les propriétés qu'on souhaite et qui est de taille supérieure à

$$(2^{\frac{n^2}{16k^2}})^k 2^{-2k} = 2^{\frac{n^2}{16k} - 2k}.$$

L'espace mémoire nécessaire vaut donc au moins $\frac{n^2}{16k} - 2k \geq \frac{n^2}{16k} - 2\alpha n$, puisque $k \leq \alpha n$. Ensuite $n \leq \frac{\alpha n}{k} n$ et donc l'espace mémoire nécessaire est au moins $\frac{n^2}{k}(\frac{1}{16} - 2\alpha^2)$.

□

**Corollaire 2.** *Soit $\epsilon > 0$ une constante. Pour tout $n$ assez grand et $k(n) = O(n^{1-\epsilon})$, un algorithme de streaming en 1 passe calculant pour tout graphe $G$ ayant $n$ sommets un couplage $M(G)$ qui vérifie $|M(G)| \geq \text{MaxMat(G)} - k(n)$ nécessite une mémoire de taille $\Omega(n^{1+\epsilon})$.*

*Démonstration.* Soit $A$ un algorithme de streaming en 1 passe calculant pour tout graphe $G$ ayant $n$ sommets un couplage $M(G)$ qui vérifie $|M(G)| \geq \text{MaxMat(G)} - k(n)$ et utilisant une mémoire $s(n)$.

Par la même réduction que celle du corollaire 1, pour résoudre le problème $\text{AppMaxMat}_l^k$, on peut appliquer l'algorithme $A$ à un graphe de taille $4l$, et obtenir un protocole utilisant $s(4l)$ bits de communication. Soit $\alpha$ tel que $\frac{1}{16} - 2\alpha^2 = \frac{1}{32}$. Puisque $k(n) = O(n^{1-\epsilon})$ et que $n = 4l$ alors $k(n) \leq \alpha l$ pour $n$ assez grand. Le théorème 4 dit alors $s(4l) \geq \frac{l^2}{32k(n)}$. Il en découle $s(n) = \Omega(n^{1+\epsilon})$.

□

### 2.1.3 Borne inférieure probabiliste

On a supposé jusqu'à présent que tous les algorithmes étaient déterministes. On peut se poser la question si ces bornes inférieures s'appliquent aux algorithmes probabilistes également. Un algorithme probabiliste est un algorithme qui utilise des bits aléatoires et qui donne une solution fausse (on supposera ici qu'une solution fausse est un couplage mais qui ne vérifie pas la condition d'approximation) avec petite probabilité $\leq \frac{1}{3}$ [1], la mémoire utilisée par un algorithme probabiliste est définie comme la mémoire maximum sur les entrées et sur les bits aléatoires. Pour montrer une borne inférieure probabiliste on considère la complexité de communication probabiliste de $\text{AppMaxMat}_n^k$.

**Théorème 5.** *Pour tout $n$ et $k < n$, un protocole probabiliste qui résout $\text{AppMaxMat}_n^k$ utilise une mémoire $\Omega(\frac{n^2}{k^2 \log n})$*

*Démonstration.* Comme dans le principe minimax de Yao [Yao77], la preuve procède en construisant à partir d'un protocole probabiliste, un protocole déterministe qui marche sur une fraction des entrées. La construction déterministe à laquelle on se ramène ici est celle qui donne une borne en $\Omega(\frac{n^2}{k^2})$ et pas la construction du théorème 4. Cette dernière utilise beaucoup de graphes de Bob différents, c'est pour cela que le passage en probabiliste semble difficile.

Soit $P$ un protocole probabiliste qui résout $\text{AppMaxMat}_n^k$ avec communication $s(n,k)$. Alice et Bob ont une chaîne commune de bits aléatoires $r$ à leur disposition, Alice, en fonction de son entrée $G$ et de $r$, envoie à Bob un message. Bob reçoit ce message, et avec son entrée $H$ sort un couplage $P(G,H,r)$. La correction du protocole assure que pour tout $G$ et $H$ la probabilité d'erreur,

$$\mathbb{P}_r(|P(G,H,r)| \leq |\text{MaxMat}(G \cup H)| - k) \leq \frac{1}{3}.$$

On construit alors un protocole $P'$ qui se trompe moins souvent. $P'$ tire $t = 2\log 2n$ chaînes aléatoires indépendantes $r_1, \ldots, r_t$ et prend le couplage le plus grand parmi $P(G,H,r_1), \ldots, P(G,H,r_t)$. Remarquons que le protocole $P'$ utilise $s(n,k) \cdot t$ bits de communication. Ainsi, pour tout $G$ et $H$,

$$\mathbb{P}_r(|P'(G,H,r)| \leq |\text{MaxMat}(G \cup H)| - k) \leq (\frac{1}{3})^t \leq \frac{1}{4n^2}.$$

[1] La constante $\frac{1}{3}$ a peu d'importance et peut être remplacée par n'importe quelle constante $\gamma < 1/2$

Pour $\mathcal{G} = \{k+1$ copies de $g | g$ biparti avec $2\frac{n}{k+1}$ sommets$\}$ de la section 2.1.2, on a alors pour tout $H$,

$$\mathbb{P}_{G\in\mathcal{G},r}(|P'(G,H,r)| \leq |\textsc{MaxMat}(G \cup H)| - k) \leq \frac{1}{4n^2}.$$

Considérons maintenant l'ensemble $\mathcal{H}$ de graphes de Bob utilisés pour différencier deux graphes de $\mathcal{G}$. Puisque chaque graphe $H$ est construit avec $k+1$ motifs d'un graphe $h$, pour compter le nombre de graphes dans $\mathcal{H}$, il suffit de compter le nombre de $h$ pour un motif. Un graphe $h$ construit à partir de $g_1$ et $g_2$ ne dépend que du choix d'une arête $(u, v)$ qui est dans la différence symétrique $g_1 \oplus g_2$ comme le montre la figure 3. Ainsi, on a $|\mathcal{H}| = (\frac{n}{k+1})^2 \leq n^2$. En majorant la probabilité de l'union par la somme des probabilités,

$$\mathbb{P}_{G\in\mathcal{G},r}(\exists H \in \mathcal{H}, |P'(G,H,r)| \leq |\textsc{MaxMat}(G \cup H)| - k) \leq n^2 \cdot \frac{1}{4n^2} = \frac{1}{4}.$$

Il existe donc $r_0$ tel que

$$\mathbb{P}_{G\in\mathcal{G}}(\exists H \in \mathcal{H}, |P'(G,H,r_0)| \leq |\textsc{MaxMat}(G \cup H)| - k) \leq \frac{1}{4}.$$

Soit $\mathcal{G}' = \{G \in \mathcal{G}, \forall H \in \mathcal{H}, |P'(G,H,r_0)| \geq |\textsc{MaxMat}(G \cup H)| - k\}$ l'ensemble des graphes sur lesquels le protocole marche pour tout $H \in \mathcal{H}$. Le protocole $P'_{r_0}$ avec la chaîne $r_0$ est alors un protocole déterministe qui fonctionne sur les graphes de $\mathcal{G}' \times \mathcal{H}$. En reprenant le même raisonnement que la section 2.1.2, et en changeant $\mathcal{G}$ par $\mathcal{G}'$, le protocole $P'_{r_0}$ utilise au moins $\log|\mathcal{G}'| \geq \log \frac{3}{4} 2^{(\frac{n}{k+1})^2} \geq (\frac{n}{k+1})^2 - 1$ bits de communication. On en déduit

$$2\log(2n) \cdot s(n,k) \geq (\frac{n}{k+1})^2 - 1.$$

Soit $s(n,k) = \Omega(\frac{n^2}{k^2 \log n})$.

□

Donc, pour le problème du couplage en streaming, on obtient une borne inférieure probabiliste un peu plus faible que la borne inférieure déterministe :

**Corollaire 3.** *Soit $\epsilon > 0$ une constante. Pour tout $n$ assez grand et $k(n) = O(n^{\frac{1}{2}-\epsilon})$, un algorithme probabiliste de streaming en 1 passe calculant pour tout graphe $G$ ayant $n$ sommets un couplage $M(G)$ qui vérifie $|M(G)| \geq \textsc{MaxMat}(\mathrm{G}) - k(n)$ nécessite une mémoire de taille $\Omega(n^{1+\epsilon})$.*

## 2.2 Borne inférieure dans un modèle restreint

Vu que démontrer une borne inférieure pour le problème d'approximation à un facteur constant dans ce modèle semble difficile, on peut tenter de montrer des bornes inférieures dans un modèle restreint ou sur une classe restreinte d'algorithmes. Par exemple, tous les algorithmes connus (l'algorithme glouton, les algorithmes dans le cas avec poids [FKM+05b, Zel08]) ne maintiennent en mémoire qu'une partie graphe en entrée. Ces algorithmes fonctionnent de la manière suivante : quand une nouvelle arête arrive, on la garde en mémoire, puis on peut retirer des arêtes qui sont stockées en mémoire. Les arêtes que l'on décide de retirer de la mémoire ne peuvent plus jamais être récupérées. Lorsque le stream est passé, un couplage maximum du graphe en mémoire est alors calculé. Il est aussi naturel de s'intéresser à cette

classe d'algorithme puisqu'il s'agit des algorithmes utilisant la technique de sparsification introduite dans [EGIN97]. L'esprit de ce que l'on cherche à montrer est que si on oublie trop d'arêtes, alors on pourra compléter le graphe de sorte que ces oublis comptent pour la taille du couplage maximum. Pour cela on considère que l'exécution de l'algorithme sur un stream qui débute par une clique de grande taille. On regarde alors le graphe maintenu en mémoire pour cette clique. Si celui-ci a trop peu d'arêtes il existe un moyen de compléter le stream de telle sorte que le stream ait un couplage beaucoup plus grand que le graphe maintenu en mémoire.

**Théorème 6.** *Soit $A$ un algorithme qui fonctionne en une passe et qui renvoie un couplage du graphe en entrée. Supposons que $A$ ne garde en mémoire qu'un sous-ensemble d'arêtes du graphe en entrée de taille au plus $l|V|$. Alors pour tout $\epsilon > 0$, il existe un graphe $G$ pour lequel* $|A(G)| \leq (\frac{4}{3+\sqrt{1+1/l}} - \epsilon) \cdot \textsc{MaxMat}(G)$.

*Démonstration.* Le graphe en entrée est un graphe biparti $(L, R)$ avec $L = L_1 \cup L_2$ et $R = R_1 \cup R_2, |L_1| = |L_2| = |R_1| = |R_2| = n$. On a alors $|V| = 4n$. La preuve procède également en considérant la répartition des arêtes introduite dans la section précédente, c'est-à-dire que la première partie du stream contient les arêtes de $G$ entre les sommets $L_1$ de gauche et $R_1$ de droite, et la deuxième partie concerne le reste des arêtes.

Soit $A$ un algorithme qui calcule un couplage en une passe, et qui maintient en mémoire un graphe dont la taille est $\leq kn$, en notant $k = 4l$. Soit alors $E'$ les arêtes en mémoire obtenues lorsque la première partie du flux est le graphe complet $E = L_1 \times R_1$ entre $L_1$ et $R_1$, ($|E'| \leq kn$).

Commençons par indiquer un schéma de la preuve. Puisque le graphe $G' = (L_1, R_1, E')$ a peu d'arêtes, on va montrer qu'il existe un ensemble $X$ de taille linéaire dont le complémentaire du voisinage dans $R_1$ $R_1 - \Gamma_{G'}(X)$ est aussi de taille linéaire. Une fois cet ensemble choisi, on va exhiber un ensemble d'arêtes $H$ pour rendre les arêtes entre $X$ et $R_1 - \Gamma_{G'}(X)$ (qui sont des arêtes qu'on a oubliées) importantes. Plus précisément, la différence entre $\textsc{MaxMat}(E \cup H)$ et $\textsc{MaxMat}(E' \cup H)$ sera linéaire en $n$.

Soit $\alpha$ un paramètre à fixer plus tard, et $Y = \{v \in L_1, deg_{G'}(v) \leq \alpha k\}$, alors $|Y| \geq (1 - \frac{1}{\alpha})n$. En effet, sinon le nombre d'arêtes dans $E'$ serait $> \frac{1}{\alpha}n \times \alpha k = kn$, ce qui contredirait la contrainte du nombre d'arêtes dans $E'$. On choisit $X \subset Y$ quelconque de taille $\lfloor (1 - \frac{1}{\alpha})n \rfloor$.

Maintenant $|\Gamma_{G'}(X)| \leq \alpha k \times |X|$, et donc le complémentaire $|R_1 - \Gamma_{G'}(X)| \geq n - \alpha k|X| \geq (1 - \alpha k + k)n$. La valeur de $\alpha$ est fixée pour que $|X| \leq |R_1 - \Gamma_{G'}(X)|$. Ceci peut être vérifié si $\alpha$ vérifie

$$1 - \frac{1}{\alpha} = 1 - \alpha k + k$$
$$\alpha^2 k - \alpha k - 1 = 0$$

Ceci donne $\alpha = \frac{1}{2}(1 + \sqrt{1 + \frac{4}{k}})$.

$H$ est ensuite construit comme indiqué dans la figure 6 : un couplage entre $L_1 - X$ et $n - |X|$ sommets de $R_2$, et un couplage entre $n - |X|$ sommets de $L_2$ et $\Gamma_{G'}(X) \cup A$ où $A$ est un ensemble de sommets éventuellement vide de $R_1 - \Gamma_{G'}(X)$ pour que $|\Gamma_{G'}(X) \cup A| = n - |X|$.

On a alors

$$\textsc{MaxMat}(E \cup H) = (n - |X|) + (n - |X|) + |X| = 2n - |X|$$

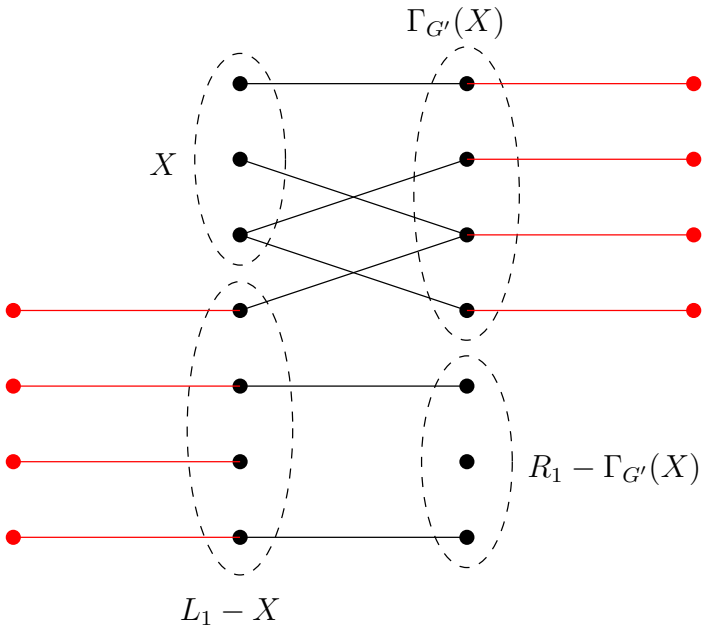


**Fig. 6** – *Construction de $H$ (en rouge) : un couplage entre $L_1 - X$ et $n - |X|$ sommets de $R_2$, et un couplage entre $n - |X|$ sommets de $L_2$ et $\Gamma_{G'}(X)$*

et

$$\textsc{MaxMat}(E' \cup H) = (n - |X|) + (n - |X|) = 2n - 2|X|.$$

Le facteur d'approximation de l'algorithme ne peut alors excéder

$$\begin{aligned}
\frac{2n - 2|X|}{2n - |X|} &= \frac{2n - 2\lfloor (1 - 1/\alpha)n \rfloor}{2n - \lfloor (1 - 1/\alpha)n \rfloor} \\
&\leq \frac{2n - 2(1 - 1/\alpha)n + 2}{2n - (1 - 1/\alpha)n} \\
&= \frac{2/\alpha}{1 + 1/\alpha} - \frac{2}{(1 + 1/\alpha)n} \\
&= \frac{2}{1 + \frac{1}{2}(1 + \sqrt{1 + \frac{4}{k}})} - \frac{2}{(1 + 1/\alpha)n} \\
&= \frac{4}{3 + \sqrt{1 + \frac{4}{k}}} - \frac{2}{(1 + 1/\alpha)n}
\end{aligned}$$

Pour $n$ assez grand $\frac{2}{(1+1/\alpha)n} \leq \epsilon$, d'où le résultat.

□

Par exemple, si ce théorème est appliqué avec $l = \frac{1}{2}$, la constante d'approximation ne peut pas excéder 0.85. Ce théorème donne bien une borne inférieure linéaire mais utilise un nombre d'arêtes uniquement linéaire, alors qu'il serait envisageable d'avoir une mémoire en $n \cdot \text{polylog}(n)$. Si $k = \log n$ dans le théorème par exemple, la borne inférieure sur le facteur d'approximation tend vers 0. On peut alors se demander si cette méthode peut s'étendre à des

graphes avec un nombre d'arêtes un peu plus que linéaire. La réponse est non : un ensemble $X$ vérifiant les propriétés demandées (à savoir un ensemble $X$ de taille linéaire tel que la taille du complémentaire du voisinage de $X$ soit aussi linéaire) n'existe pas pour tout graphe contenant moins de $kn$ arêtes. En fait, presque tous les graphes qui ont $n \log n$ arêtes ne vérifient pas cette propriété.

**Lemme 5.** *Soit $\epsilon > 0$, et $G$ un graphe biparti aléatoire sur les sommets $(L, R)$ ayant $n \log n$ arêtes, alors*

$$\mathbb{P}(\exists X \subset L, |X| \geq \epsilon n, \text{ avec } |R - \Gamma(X)| \geq \epsilon n) \xrightarrow[n\to\infty]{} 0$$

*Démonstration.* Le preuve de ce lemme est un cas particulier de la preuve du lemme 7 de la section suivante. Il suffit de lire la preuve du lemme 7 en prenant comme graphe $G$ du lemme 7 le graphe biparti complet. □

Ceci nous amène à l'idée de considérer l'algorithme très simple suivant : on prend chaque arête avec probabilité $\frac{n \log n}{m}$, et à la fin on calcule un couplage maximum du graphe en mémoire qui a $n \log n$ arêtes en moyenne. Cet algorithme trouve pour le graphe complet biparti pour tout $\epsilon > 0$ un couplage de taille $(1-\epsilon)n$ avec grande probabilité (dépendant de $\epsilon$). Mais lorsque le graphe présente moins de régularité, l'algorithme ne peut pas fonctionner tel quel vu que certains sommets peuvent avoir peu de voisins et donc vont se retrouver isolés avec grande probabilité. Mais il est quand même possible de maintenir un degré minimum, par exemple chaque sommet choisit au hasard $\log n$ voisins. Il se trouve que cet algorithme ne marche pas non plus, par exemple sur le graphe de la figure 7. En effet, si ce sont les arêtes noires qui arrivent en premier, en moyenne $O(\log n)$ arêtes parmi les arêtes en gras vont être maintenues par l'algorithme. Puis lorsque les arêtes en rouge arrivent, on se rend compte que les arêtes en gras étaient importantes, mais puisqu'on en a oublié une grosse partie, on ne peut pas sortir un grand couplage.

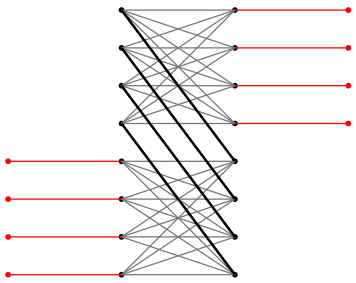

**Fig. 7** – *Un échantillonnage simple ne peut pas différencier les arêtes en gras*

## 2.3 Algorithme dans un cas particulier

Cette discussion nous montre que la "régularité" du graphe est une caractéristique qui rendrait peut être plus facile nos analyses. Il semble qu'une certaine régularité ferait qu'un sous-graphe aléatoire du graphe initial représenterait bien le graphe. Une notion de régularité utilisée est la suivante :

**Definition 6** ($\epsilon$-régularité)**.** Soit $\epsilon > 0$. Étant donnés deux ensembles de sommets $A$ et $B$ d'un graphe $G = (V, E)$, on dit que la paire $(A, B)$ est $\epsilon$-régulière si pour tout $X \subset A, Y \subset B$ avec $|X| > \epsilon|A|$ et $|Y| > \epsilon|B|$ alors

$$|d(X,Y) - d(A,B)| < \epsilon$$

où $d(X,Y)$ représente la densité d'arêtes entre $X$ et $Y$, $d(X,Y) = \frac{e(X,Y)}{|X||Y|}$ avec $e(X,Y) = |E \cap X \times Y|$.

Cette notion est importante car elle intervient dans le célèbre lemme de régularité de Szemerédi très utilisé en théorie des graphes. Ce lemme dit que tout graphe peut être partitionné de sorte que presque toutes les paires soient $\epsilon$-régulières.

**Théorème 7** (Lemme de régularité de Szemerédi [Sze76]). *Pour tout $\epsilon > 0$ et $l$, il existe deux entiers $N(\epsilon, l)$ et $M(\epsilon, l)$ tels que pour tout graphe $G$ avec $n \geq N(\epsilon, l)$ sommets, il existe une partition des sommets en $k+1$ classes $V = V_0 \cup V_1 \cup \cdots \cup V_k$*

- $l \leq k \leq M(\epsilon, l)$
- $|V_0| < \epsilon n$
- $|V_1| = |V_2| = \cdots = |V_k|$
- *toutes les paires $(V_i, V_j)$ sont $\epsilon$-régulières sauf au plus $\epsilon k^2$.*

Ce théorème n'a d'intérêt que pour les graphes denses (i.e. qui ont $\Omega(n^2)$ arêtes), puisque pour une famille de graphes avec $G_n$ avec $o(n^2)$ arêtes, le graphe vide approxime bien $G_n$.

Tout d'abord montrons qu'une paire $(A, B)$ $\epsilon$-régulière a un couplage (entre $A$ et $B$) de taille $\geq (1-\epsilon)n$.

Pour cela, le lemme suivant est utile.

**Lemme 6.** *Soit $(A, B)$ une paire $\epsilon$-régulière avec $\epsilon < d(A, B)$. S'il n'y a pas d'arêtes entre $X \subset A$ et $Y \subset B$ alors $|X| \leq \epsilon n$ ou $|Y| \leq \epsilon n$.*

*Démonstration.* Si $X > \epsilon n$ et $Y > \epsilon n$, alors par régularité, $|d(X,Y) - d(A,B)| < \epsilon$. Mais $d(X,Y) = 0$, et donc $d(A,B) < \epsilon$ ce qui contredit l'hypothèse de départ. □

**Propriété 2.** *Soit $G = (L, R, E)$ un graphe biparti avec $|L| = |R| = n$, et $\epsilon < \frac{1}{2}$, et $\epsilon < d(L,R) = \frac{|E|}{|L||R|}$. Si $(L,R)$ est $\epsilon$-régulière, alors $G$ a un couplage de taille $\geq (1-\epsilon)n$.*

*Démonstration.* Pour cela, on montre que la deficency du graphe $G = (L, R, E)$ est plus petite que $\epsilon n$, ce qui montre par le théorème de Hall que $G$ a un couplage de taille $\geq (1-\epsilon)n$.

Pour $X \subset L$ tel que $|X| \leq \epsilon n$, on a $def(X) = |X| - |\Gamma(X)| \leq \epsilon n$.

Pour $X \subset L$ tel que $|X| > \epsilon n$, on peut appliquer le lemme 6 au non voisinage de $X$. On en déduit que celui-ci est de taille $\leq \epsilon n$ et donc que le voisinage $\Gamma(X)$ de $X$ vérifie $|\Gamma(X)| \geq (1-\epsilon)n$. Finalement $def(X) = |X| - |\Gamma(X)| \leq n - (1-\epsilon)n \leq \epsilon n$.

En conclusion $def(G) \leq \epsilon n$. □

L'approximation du couplage maximum entre des paires $\epsilon$-régulières est facile dans la mesure où un sous-graphe aléatoire de petite taille avec $n \log n$ arêtes conserve un couplage maximum de grande taille. En fait, on utilise une propriété un peu plus générale que la $\epsilon$-régularité.

**Propriété 3.** *Soit $G = (L, R, E)$ un graphe biparti avec $|L| = |R| = n$, et $\epsilon < d = d(L,R) = \frac{|E|}{|L||R|}$. On suppose que $(L,R)$ est $\epsilon$-régulière. Alors il existe $\delta > 0$ tel que pour tout $X \subset L$, $Y \subset R$ avec $|X| > \epsilon n$, $|Y| > \epsilon n$, $d(X,Y) > \delta$. On dira que $(L,R)$ est $(\epsilon, \delta)$ dense.*

*Démonstration.* Il suffit de prendre $\delta = d - \epsilon$. □

**Lemme 7.** *Soit $\epsilon, \delta > 0$. Soit $G = (L, R, E)$ un graphe biparti avec $|L| = |R| = n$, où $(L, R)$ est une paire $(\epsilon, \delta)$ dense. Alors un sous-graphe aléatoire de $G$ contenant $n \log n$ arêtes a un couplage de taille $\geq (1 - \epsilon)n$ avec grande probabilité.*

*Démonstration.* L'analyse est plus simple si on considère un sous-graphe aléatoire où chaque arête est choisie avec probabilité $p$. Soit $g$ un sous-graphe aléatoire de $G$ où chaque arête est prise avec probabilité $p$ (on notera $g \in \mathcal{G}_p$). Pour cela, on borne la probabilité que la deficiency du graphe $g$ soit plus grande que $\epsilon n$. $\Gamma(X)$ représente le voisinage de $X \subset L$ dans le graphe aléatoire $g$.

$$\begin{aligned}
\mathbb{P}_g(def(g) > \epsilon n) &\leq \mathbb{P}_g(\exists S \subset L, |S| - |\Gamma(S)| > \epsilon n) \\
&\leq \sum_{S \subset L, |S| > \epsilon n} \mathbb{P}_g(|\Gamma(S)| \leq |S| - \epsilon n) \\
&\leq \sum_{S \subset L, |S| > \epsilon n} \sum_{T \subset R, |T| < |S| - \epsilon n} \mathbb{P}_g(\Gamma(S) \subset T)
\end{aligned}$$

Maintenant pour un $S$ tel que $|S| > \epsilon n$ et $T$ tel que $|R - T| > \epsilon n$. La condition de densité s'applique alors entre les ensembles $S$ et $R - T$, d'où la densité $d(S, R - T) \geq \delta$, et donc le nombre d'arêtes entre $S$ et $R - T$ dans le graphe $G$ est au moins $\delta(\epsilon n)^2$.

$$\begin{aligned}
\mathbb{P}_g(\Gamma(S) \subset T) &\leq (1 - p)^{e(S, R-T)} \\
&\leq (1 - p)^{\delta \epsilon^2 n^2}
\end{aligned}$$

Il en découle que

$$\begin{aligned}
\mathbb{P}_g(def(g) > \epsilon n) &\leq 2^{2n}(1 - p)^{\delta \epsilon^2 n^2} \\
&\leq 2^{2n} e^{-p\delta \epsilon^2 n^2} \\
&= e^{2 \ln 2n - p\delta \epsilon^2 n^2}
\end{aligned}$$

Donc si $p = \frac{\log n}{n}$ par exemple, cette probabilité tend vers 0 lorsque $n$ tend vers l'infini. Plus généralement, il suffit que $p > \frac{2 \ln 2}{\delta \epsilon^2} \frac{1}{n}$.

Montrons maintenant que cette propriété reste vraie si le sous-graphe aléatoire est choisi avec un nombre fixé d'arêtes $M = n \log n$. On montre d'abord qu'avec grande probabilité un $g \in \mathcal{G}_p$ a moins de $n \log n$ arêtes. Pour $p = \frac{4 \ln 2}{\delta \epsilon^2} \frac{1}{n}$, et $g \in \mathcal{G}_p$, par l'inégalité de Markov

$$\mathbb{P}_{g \in \mathcal{G}_p}(e(g) > n \log n) \leq \frac{\mathbb{E}(e(g))}{n \log n} = \frac{pn^2}{n \log n} = O(\frac{1}{\log n})$$

en notant $e(g)$ le nombre d'arêtes de $g$, et le $O(.)$ ne dépend que de $\epsilon$ et de $\delta$. D'autre part pour $g \in \mathcal{G}_p$, sachant que $e(g) \leq M$, $g$ est distribué uniformément sur l'ensemble des graphes ayant moins de $M$ arêtes. On notera $g \in \mathcal{G}_{\leq M}$ et $g \in \mathcal{G}_M$ lorsque $g$ est distribué uniformément sur les sous-graphes de $G$ ayant moins de $M$ ou exactement $M$ arêtes respectivement.

Si on note $P(g)$ le fait que $g$ a un couplage de taille $\geq (1-\epsilon)n$, on a

$$\begin{aligned}\mathbb{P}_{g\in\mathcal{G}_p}(P(g)) &= \mathbb{P}_{g\in\mathcal{G}_{\leq M}}(P(g))\mathbb{P}_{g\in\mathcal{G}_p}(e(g)\leq n\log n)\\ &\quad + \mathbb{P}_{g\in\mathcal{G}_p}(P(g)|e(g)>n\log n)\mathbb{P}_{g\in\mathcal{G}_p}(e(g)>n\log n)\\ &\leq \mathbb{P}_{g\in\mathcal{G}_{\leq M}}(P(g)) + \mathbb{P}_{g\in\mathcal{G}_p}(e(g)>n\log n).\end{aligned}$$

Mais puisque l'ajout d'arêtes ne peut qu'augmenter les chances de vérifier $P$, $\mathbb{P}_{g\in\mathcal{G}_M}(P(g)) \geq \mathbb{P}_{g\in\mathcal{G}_{\leq M}}(P(g))$. Pour mieux voir cette inégalité, il suffit de dire que pour générer un $g\in\mathcal{G}_M$, il est possible de générer un $h\in\mathcal{G}_{\leq M}$ puis de le compléter par un graphe $h'$ aléatoire avec le bon nombre d'arêtes, et alors $\mathbb{P}_g(P(g)) = \mathbb{P}_{h,h'}(P(h\cup h')) \leq \mathbb{P}_h(P(h))$.

Ainsi,

$$\begin{aligned}\mathbb{P}_{g\in\mathcal{G}_M}(P(g)) &\geq \mathbb{P}_{g\in\mathcal{G}_p}(P(g)) - O(\frac{1}{\log n})\\ &\geq 1 - e^{-2\ln 2n} - O(\frac{1}{\log n})\\ &= 1 - O(\frac{1}{\log n})\end{aligned}$$

Donc avec probabilité tendant vers 1 lorsque $n\to\infty$, le couplage maximum de sous-graphe aléatoire est de taille supérieure ou égale à $(1-\epsilon)n$.

□

Cette propriété donne donc immédiatement un algorithme de streaming qui calcule une $(1-\epsilon)$ approximation pour les paires $\epsilon$-régulières denses ou plus généralement pour les paires $(\epsilon,\delta)$ denses, il suffit de maintenir un échantillon aléatoire des arêtes de taille $n\log n$, ceci peut être fait avec un échantillonnage réservoir.

**Algorithme d'échantillonage :**

1. Pour la $i$-ème arête $e$,
   - si $i\leq n\log n$ garder l'arête en mémoire
   - si $i>n\log n$, avec probabilité $\frac{1}{i}$, garder $e$ en mémoire en prenant la place d'une arête en mémoire choisie au hasard.
2. Lorsque tout le graphe est lu, calculer un couplage maximum du graphe en mémoire (qui a $n\log n$ arêtes).

Mais sur une paire $\epsilon$-régulière, l'algorithme glouton marche bien aussi.

**Propriété 4.** *Soit $\epsilon,\delta>0$. Soit $G=(L,R,E)$ un graphe biparti. Si $(L,R)$ est une paire $(\epsilon,\delta)$ dense, alors l'algorithme glouton sur $G$ trouve un couplage de taille $\geq(1-\epsilon)n$.*

*Démonstration.* Il suffit de voir qu'un couplage maximal a nécessairement une taille $\geq(1-\epsilon)n$. Pour cela, considérons les sommets non couplés $X$ et $Y$ de part et d'autre du graphe. Alors nécessairement, il n'y a aucune arête entre $X$ et $Y$, sinon l'algorithme glouton les aurait ajouté au couplage. Il en découle alors, $|X|\leq\epsilon n$ ou $|Y|\leq\epsilon n$ (sinon on aurait $d(X,Y)\geq\delta$) et puisque $|L|=|R|$, alors $|X|=|Y|$ et donc $|X|=|Y|\leq\epsilon n$. □

# 3 Couplage maximum dans les modèles à plusieurs passes

On étudie maintenant le problème lorsque plusieurs passes sur l'entrée sont possibles. Un algorithme peut donc utiliser les informations acquises et stockées en mémoire lors d'une passe dans une passe ultérieure.

## 3.1 Modèle lecture seule

On se place ici dans le modèle semi-streaming, c'est-à-dire que l'on dispose d'une mémoire de taille $O(n \cdot \text{polylog}(n))$. Il est possible de calculer un couplage maximal en une passe. Les articles [FKM+05b, McG05] présentent des algorithmes qui améliorent à chaque passe le couplage courant en trouvant des chemins augmentants. L'algorithme de [FKM+05b] donne une approximation à $\frac{2}{3} - \epsilon$ près du couplage maximum dans un graphe biparti en $O(\frac{\log(1/\epsilon)}{\epsilon})$ passes. On donne une description assez brève de l'algorithme pour donner l'esprit d'un algorithme en plusieurs passes en annexe.

## 3.2 Modèle lecture/écriture

Un autre modèle étudié dans la littérature est le modèle de stream en lecture/écriture. Ce modèle se rapproche des modèles de rubans qui étaient étudiés dans les années 40-60 motivés par les limitations des disques de l'époque.

Le stream passe alors un petit nombre de fois mais à chaque passe un stream de sortie est écrit sur un autre stream, et celui-ci est utilisé comme entrée pour la passe suivante. La taille du stream ne doit cependant pas trop augmenter, pour cela on impose que la taille du stream reste linéaire. Dans ce modèle, la sortie peut se présenter sur le stream lui-même, on peut donc imaginer résoudre le problème du couplage maximum ou l'approximer avec une mémoire sous-linéaire en le nombre de sommets avec un nombre constant de passes par exemple. Ce n'est en fait pas possible à cause du théorème suivant.

**Théorème 8.** *Si un algorithme probabiliste dans le modèle de streaming en lecture/écriture calcule un couplage maximal pour des graphes ayant $n$ sommets en $p$ passes avec une mémoire $s$ alors $p \cdot s = \Omega(n)$.*

*Démonstration.* La difficulté du problème du couplage maximal découle de la difficulté du problème de communication de Disj. En effet, avec un algorithme utilisant $p$ passes et une mémoire $s$ pour le couplage de graphe avec $n$ sommets, on peut résoudre le problème Disj de taille $n$ avec $O(ps)$ bits de communication.

Soit $x = \{x_1, \ldots, x_a\}$ et $y = \{y_1, \ldots, y_b\}$ les ensembles d'entrée d'Alice et de Bob. On construit un graphe biparti $G = (L, R, E)$ avec $L = \{1, \ldots, 2n\}$ et $R = \{1, \ldots, n\}$, contenant les arêtes suivantes :

- $E_A = \{(i, x_i), i \in \{1, \ldots a\}\}$, les arêtes d'Alice
- $E_B = \{(n + j, y_j), j \in \{1, \ldots b\}$, les arêtes de Bob

Le protocole de communication est alors le suivant. Alice fait tourner l'algorithme sur $E_A$, garde l'état du stream, envoie l'état de la mémoire à Bob, qui continue d'exécuter l'algorithme puis fait de même en renvoyant son état de la mémoire à Alice, qui continue l'algorithme. Lorsque l'algorithme est terminé, il est possible de calculer le nombre d'arêtes dans le couplage avec une communication $O(\log n)$ (chacun envoie le nombre d'arête dans la partie du couplage qu'il détient). De plus Alice et Bob s'envoient les tailles de leurs ensembles $a$ et $b$. Pour décider si les ensembles $x$ et $y$ sont disjoints, il suffit de comparer $a+b$ et la taille du couplage renvoyé.

Notre protocole a donc un coût en communication de $2ps + O(\log n)$. Sachant que la complexité de communication du problème Disj est $\Omega(n)$ d'après [KS92], on a $ps = \Omega(n)$. □

En réalité la possibilité d'écrire sur le flux n'augmente pas vraiment la puissance de calcul, par exemple, pour les problèmes de décision, il suffit d'augmenter la mémoire d'un facteur $p$ pour simuler la possibilité d'écrire.

**Lemme 8** ([Ruh03]). *Pour un problème de décision, un algorithme de streaming en lecture/écriture en p passes avec mémoire s peut être transformé en un algorithme de streaming simple (en lecture seule) en p passes avec mémoire sp.*

## 3.3 Modèle de streaming avec procédure de tri

### 3.3.1 Motivations, modèle

Ce n'est donc pas tant le fait de ne pas pouvoir écrire qui limite le modèle de streaming, mais plutôt l'impossibilité de réordonner le stream. En effet, pour trier une liste de $n$ éléments dans le modèle de streaming avec écriture, alors il faut que $sp \geq n$ [Ruh03]. Pourtant des machines qui trient extrêmement rapidement existent [Aga96], elles atteignent des vitesses comparables aux vitesses des accès mémoire. Pour prendre ceci en compte [ADRR04] définissent un modèle "streaming and sorting" qui contient une primitive de tri du stream. Le tri est effectué par une fonction de comparaison qui tient dans la mémoire. Une passe de tri prend donc en entrée un stream et renvoie le stream trié selon la fonction de comparaison en mémoire.

Un autre motivation pour ce modèle est le fait que le tri est une opération qui peut être effectuée dans des modèles de streaming avec plusieurs rubans (modèle introduit dans [GS05] par exemple). En effet, une machine de Turing à 3 rubans peut trier une liste de $n$ éléments en $O(\log n)$ passes et avec mémoire $O(\log n)$. Le principe est basé sur le tri fusion, la première passe trie les éléments 2 par 2, la deuxième 4 par 4 et ainsi de suite jusqu'à ce que la liste entière soit triée. Deux rubans servent pour lire les deux listes à fusionner et le troisième pour écrire la liste fusionnée.

**Definition 7** (**StrSort** [Ruh03]). On définit $\mathbf{StrSort}(p_{Str}, p_{Sort}, m)$ l'ensemble des fonctions calculables en composant $p_{Str}$ passes en lecture écriture, $p_{Sort}$ passes de tri et une mémoire de taille $m$. De plus, la taille des streams intermédiaires doit rester linéaire.

On s'intéressera surtout à $\mathbf{StrSort}(O(\text{polylog}(n)), O(\text{polylog}(n)), O(\text{polylog}(n)))$, que l'on notera simplement **StrSort**. Lorsqu'on parlera du nombre de passes d'un algorithme, il s'agira de la somme du nombre de passes normales et du nombre de passes de tri. Pour une description exhaustive du modèle, se référer à [Ruh03].

Les algorithmes dans ce modèle ressemblent aux algorithmes parallèles dans les modèles PRAM. Le modèle PRAM est un modèle dans lequel plusieurs processeurs ont accès à une mémoire partagée, le but est alors de minimiser le temps de calcul des processeurs. Vu que plusieurs processeurs peuvent écrire en même temps sur une même case mémoire, il existe plusieurs variantes du modèle dans lequel les lectures et écritures simultanées sont autorisées ou pas. Il s'avère que ces modèles peuvent se simuler les uns les autres au prix d'une petite perte en temps d'exécution (un facteur logarithmique), on se limitera au modèle où les lectures et les écritures simultanées sont autorisées, et les conflits écritures simultanées en donnant à chaque processeur une priorité (modèle PRIORITY PRAM).

La classe **NC** est l'ensemble des problèmes qui sont massivement parallélisables, dans le sens où avec un nombre polynomial de processeurs, le problème peut être résolu en temps poly-logarithmique.

**Definition 8.** La classe **NC** est l'ensemble des problèmes pour lesquels il existe un algorithme en temps poly-logarithmique utilisant un nombre polynomial de processeurs.

La classe **NC** est aussi l'ensemble des problèmes résolubles par une famille uniforme de circuit de taille polynomiale et de profondeur poly-logarithmique.

Avec la vision circuit de la classe **NC**, il est possible de voir un lien avec la classe **StrSort**. Les passes de tri correspondent à l'arrangement des sorties d'un certain niveau dans l'ordre pour correspondre à l'ordre des portes du prochain niveau.

**Lemme 9** ([ADRR04]). *Une fonction dans **NC** calculable avec un nombre linéaire de processeurs est aussi dans **StrSort**.*

*Démonstration.* On donne l'idée de la preuve. On montre qu'une famille uniforme de circuits peut être évaluée dans **StrSort**. L'algorithme calcule et maintient sur le stream pour chaque profondeur $l$ du circuit la liste des entrées des portes du circuit de cette profondeur dans l'ordre des portes. Une passe calcule les sorties de chaque porte. Pour passer aux entrées de la profondeur $l+1$, il suffit de marquer les sorties avec la porte qui les prend en tant qu'entrée et de les trier dans l'ordre voulu. □

Ainsi, dans ce modèle de streaming, des algorithmes parallèles efficaces existant pour le problème du couplage maximum peuvent être traduits directement. Le problème du calcul d'un couplage maximum dans la classe **RNC**, qui est la classe **NC** randomisée, mais les algorithmes connus, par exemple [KUW86], utilisent un trop grand nombre de processeurs $O(n^{6.5})$. Ces algorithmes sont basés sur une caractérisation algébrique des couplages parfaits dans les graphes à l'aide de déterminants.

### 3.3.2 Couplage maximal

Remarquons qu'avec $O(\text{polylog}(n))$ mémoire, il n'est pas évident de trouver un couplage maximal. En effet, garder les sommets qui sont déjà couplés est impossible. Donc il n'est pas clair que calculer un couplage maximal est possible dans le modèle **StrSort**. Mais il s'avère qu'il est possible de trouver un couplage maximal en utilisant un algorithme randomisé en temps moyen $O(\log n)$ avec $O(|E|)$ processeurs [II86].

**Théorème 9** ([II86]). *Il existe un algorithme randomisé qui calcule un couplage maximal en un nombre de passes moyen de $O(\log n)$ dans le modèle streaming and sorting.*

Il existe également un algorithme déterministe dans le modèle PRAM utilisant un nombre de processeurs $O(|E|)$ en temps $O(\log^3 |V|)$ dans [IS86]. Calculer un couplage maximal est donc dans **StrSort** de manière déterministe.

### 3.3.3 Approximations du couplage maximum

Des approximations du couplage maximum dans les modèles parallèles ne sont pas très étudiées dans la littérature, ce qui est compréhensible puisque depuis [KUW86], un algorithme qui résout le problème exactement est connu, et que la classe **NC** avec un nombre linéaire de processeurs ne semblait pas une classe de complexité d'une grande importance.

Cependant, l'algorithme d'approximation dans [FKM+05b] décrit en annexe qui donne dans le modèle semi-streaming une approximation à $\frac{2}{3} - \epsilon$ près du couplage maximum dans un graphe biparti en $O(\frac{\log 1/\epsilon}{\epsilon})$ passes, peut être traduit dans un modèle PRAM. La principale difficulté est de trouver un couplage maximal entre certains sommets, ce qui peut être fait efficacement d'après [IS86].

On décrit sans détailler l'algorithme dans le modèle PRAM. L'algorithme utilise $|E|$ processeurs (sans compter les processeurs qui exécutent l'algorithme de [IS86]) qui gèrent chacune des arêtes. L'état de chaque sommet est maintenu dans un tableau en mémoire. L'algorithme qui trouve un grand ensemble de chemins augmentants se traduit ainsi :

- Répéter de manière séquentielle
  - Trouver un couplage maximal (avec [IS86]) entre des sommets libres de $R$ et des sommets couplés de $L$.
  - Marquer les sommets couplés (les sommets qui ont une aile gauche).
  - Calculer la taille de ce couplage (se fait en $O(\log n)$ étapes avec un arbre). Si la taille $\leq \delta|M|$, quitter.
  - Trouver un couplage maximal (avec [IS86]) entre les sommets de $R$ qui sont couplés dans $M$ à des sommets de $L$ qui ont une aile gauche.
  - Oublier les sommets d'une arête de $M$ qui a une aile gauche.
  - Oublier les sommets des ailes des arêtes de $M$ qui ont deux ailes (celles-ci vont être utilisées dans les chemins augmentants).

De plus toutes les opérations d'augmentations du couplage peuvent se faire en parallèle. Il en découle donc un algorithme dans le modèle streaming and sorting qui calcule une $\frac{2}{3} - \epsilon$ approximation du couplage maximum.

### 3.3.4 Couplage maximum avec poids

Il semble que le meilleur algorithme PRAM utilisant un nombre linéaire de processeurs et résolvant le couplage maximum avec poids est l'algorithme de [UC00].

**Théorème 10.** *[UC00] Il existe un algorithme qui approxime le couplage de poids maximal à un facteur $\frac{1}{2+\epsilon}$ en $O(\frac{1}{\epsilon}\log^5 n)$ passes dans le modèle streaming and sorting.*

# 4 Conclusion

Dans ce rapport, le problème de couplage maximum a été étudié dans divers modèles de streaming. Dans le modèle de streaming à une passe, le meilleur algorithme existant est l'algorithme glouton le plus simple. La contrainte d'espace empêche-t-elle de faire mieux ? La borne inférieure du théorème 4 fait un pas dans ce sens en montrant qu'il n'existe pas d'algorithme dans ce modèle simple de semi-streaming qui approche la taille du couplage maximum avec un facteur additif de $O(n^{1-\epsilon})$ pour $\epsilon > 0$. Mais ceci laisse la possibilité de l'existence d'une meilleure approximation. Il n'est pas clair si la difficulté est une difficulté de communication ou pas. En effet, la discussion de 2.2 suggère que le problème de communication peut être facile. Existe-t-il un protocole de communication pour le problème d'approximation à $(1-\epsilon)$ près avec communication $O(n \cdot \text{polylog}(n))$ ? Dans ce cas, la réduction à un problème de communication ne pourrait pas donner de meilleures bornes inférieures. Un modèle de communication avec plusieurs parties permettrait-il d'obtenir de meilleures bornes inférieures ?

**Problème ouvert 1.** *Montrer qu'un algorithme en une passe qui trouve un couplage de taille supérieure à $\frac{1}{2} + \epsilon$ avec $\epsilon > 0$ nécessite une mémoire de taille $\Omega(n^2)$, ou bien trouver un tel algorithme qui utilise $o(n^2)$ mémoire.*

La performance de l'algorithme est mesurée dans le pire cas sur l'ordre d'arrivée des arêtes. Il est possible de s'intéresser à des problèmes dans lesquels l'ordre d'arrivée des arêtes n'est

pas arbitraire. Par exemple, dans le cas d'un graphe biparti, si les arêtes arrivent par sommet, à la manière d'une liste d'adjacence, un algorithme online [KVV90] qui atteint un facteur d'approximation en moyenne de $\frac{e-1}{e}$ peut être adapté dans le modèle semi-streaming à une passe.

Si le modèle autorise plusieurs passes sur l'entrée, des algorithmes meilleurs que l'algorithme glouton existent. Mais aucune borne inférieure n'est connue, aucune borne inférieure n'élimine l'existence d'un algorithme semi-streaming même pour le calcul exact avec 2 passes. La technique généralement utilisée pour prouver des bornes inférieures avec plusieurs passes est l'utilisation de la complexité de communication à deux sens comme dans la preuve du théorème 8. Mais la complexité de communication à deux sens du couplage parfait semble être un problème difficile. Ce problème est posé comme une question ouverte dans [HMT88] et semble l'être encore aujourd'hui. En fait, dans [HMT88], le modèle est plus difficile que le nôtre car la répartition des arêtes entre Alice et Bob est prise dans le meilleur des cas, alors que nous ne voulons qu'une borne inférieure pour une répartition donnée. Néanmoins, le problème reste assez difficile.

**Problème ouvert 2.** *Montrer qu'un algorithme semi-streaming en 2 passes calculant un couplage de taille maximale ne peut pas exister, ou en trouver un.*

Je remercie toute l'équipe Algorithmique et Complexité du LRI pour son accueil et en particulier Frédéric Magniez pour son aide tout au long du stage.

## Références

[ADRR04] Gagan Aggarwal, Mayur Datar, Sridhar Rajagopalan, and Matthias Ruhl. On the streaming model augmented with a sorting primitive. In *IEEE Symposium on Foundations of Computer Science*, pages 540–549, 2004.

[Aga96] Ramesh C. Agarwal. A super scalar sort algorithm for RISC processors. In *ACM International Conference on Management of Data*, pages 240–246, 1996.

[AMS99] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1) :137–147, 1999.

[BYKS02] Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 623–632, 2002.

[EGIN97] David Eppstein, Zvi Galil, Giuseppe F. Italiano, and Amnon Nissenzweig. Sparsification a technique for speeding up dynamic graph algorithms. *Journal of the ACM*, 44 :669–696, 1997.

[FKM+05a] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the streaming model : the value of space. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 745–754, 2005.

[FKM+05b] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2-3) :207–216, 2005.

[FM83] P. Flajolet and G. N. Martin. Probabilistic counting. In *IEEE Symposium on Foundations of Computer Science*, pages 76–82, 1983.

[Gab90] Harold N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 434–443, 1990.

[GS05] Martin Grohe and Nicole Schweikardt. Lower bounds for sorting with few random accesses to external memory. In *PODS*, pages 238–249, 2005.

[HMT88] András Hajnal, Wolfgang Maass, and György Turán. On the communication complexity of graph properties. In *ACM Symposium on Theory of Computing*, pages 186–191, New York, NY, USA, 1988. ACM.

[II86] Amos Israeli and Alon Itai. A fast and simple randomized parallel algorithm for maximal matching. *Information Processing Letters*, 22(2) :77–80, 1986.

[IS86] Amos Israeli and Yossi Shiloach. An improved parallel algorithm for maximal matching. *Inf. Process. Lett.*, 22(2) :57–60, 1986.

[KS92] B. Kalayanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection. *SIAM J. on Discrete Mathematics*, 5 :545–557, 1992.

[KUW86] Richard Karp, Eliezer Upfal, and Avi Wigderson. Constructing a perfect matching is in random NC. *Combinatorica*, 6 :35–48, 1 1986.

[KVV90] R. Karp, U. Vazirani, and V. Vazirani. An optimal online bipartite matching algorithm. In *ACM Symposium on Theory of Computing*, pages 352–358, 1990.

[LP86] L. Lovász and M. D. Plummer. *Matching Theory*, volume 29 of *Annals of Discrete Mathematics*. North-Holland Math. Studies, 1986.

[McG05] Andrew McGregor. Finding graph matchings in data streams. In *APPROX-RANDOM*, pages 170–181, 2005.

[Mut06] S. Muthukrishnan. *Data streams : Algorithms and applications*. Now Publishers, 2006.

[MV80] Silvio Micali and Vijay V. Vazirani. An O(sqrt(|v|) |E|) algorithm for finding maximum matching in general graphs. In *IEEE Symposium on Foundations of Computer Science*, pages 17–27, 1980.

[Ruh03] Matthias Ruhl. *Efficient Algorithms for New Computational Models*. PhD Thesis in Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2003.

[Sze76] E. Szemerédi. Regular partitions of graphs. In *Colloques Internationaux C.N.R.S. 260, Problèmes Combinatoires et Théorie des Graphes*, pages 399–401, 1976.

[UC00] Ryuhei Uehara and Zhi-Zhong Chen. Parallel approximation algorithms for maximum weighted matching in general graphs. 76 :13–17, 2000.

[Yao77] A. C. Yao. Probabilistic computations : Toward a unified measure of complexity. In *IEEE Symposium on Foundations of Computer Science*, pages 222–227, 1977.

[Yao79] A. C. Yao. Some complexity questions related to distributed computing. In *ACM Symposium on Theory of Computing*, pages 209–213, 1979.

[Zel08] Mariano Zelke. Weighted matching in the semi-streaming model. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 669–680, 2008.

# Annexe

## Description de l'algorithme de [FKM+05b]

La méthode générale des algorithmes d'approximations pour le problème du couplage maximum est de trouver des chemins augmentants de petite taille. Ces algorithmes utilisent le fait suivant :

**Propriété 5.** *Soit $k \geq 1$. Si un couplage $M$ de $G$ n'admet pas de chemin augmentant de longueur $\leq 2k+1$ alors $|M| \geq \frac{k+1}{k+2}\textsc{MaxMat}(G)$.*

*Démonstration.* Cette propriété peut être visualisée simplement en considérant le graphe de la différence symétrique $M \oplus M^*$ où $M^*$ est un couplage maximum. En effet, le graphe $M \oplus M^*$ est un ensemble de chemins augmentants pour $M$. Le fait qu'il n'y ait pas de chemin augmentant de longueur $\leq 2k+1$ assure que les composantes connexes de $M \oplus M^*$ qui sont des chemins de longueur plus petite que $2k+1$ ont un même nombre d'arêtes qui viennent de $M$ que d'arêtes qui viennent de $M^*$. Donc au mieux, pour obtenir un couplage maximum à partir de $M$, on utilise des chemins augmentants de longueur $2k+3$, on remplace alors $k+1$ arêtes de $M$ par $k+2$ arêtes. Ce qui donne $|M| \geq \frac{k+1}{k+2}|M^*|$. □

Cet algorithme procède en phases pour trouver un grand ensemble de chemins augmentants de longueur 3, puis d'augmenter le couplage. Le fait que l'ensemble soit grand, fait que l'on s'approche en peu de phases de ne plus avoir de chemins augmentants de longueur 3. Un ensemble maximal de chemins augmentants de longueur 3 convient, mais ce n'est pas facile à trouver en streaming. L'ensemble va alors être un ensemble presque maximal de chemins augmentants pour notre couplage courant.

L'algorithme suivant permet de trouver un ensemble presque maximal de chemins simultanément augmentants de longueur 3, en entrée $G = (L, R, E)$ et un couplage maximal $M$ en mémoire.

- Répéter
  - Trouver un couplage maximal entre des sommets libres de $R$ et des sommets couplés de $L$. On appellera une arête de ce couplage une aile gauche. Si cet ensemble est de taille $\leq \delta|M|$, quitter.
  - Trouver un couplage maximal entre les sommets de $R$ qui sont couplés dans $M$ à des sommets de $L$ qui ont une aile gauche. Ce sont les ailes droites.
  - Oublier les sommets d'une arête de $M$ qui a une aile gauche (pour un tel sommet, soit une aile droite a été trouvée, soit une aile droite n'a pas été trouvée et dans ce cas on veut tout de même oublier le sommet pour laisser une place à d'autres sommets pour la prochaine passe).
  - Oublier les sommets des ailes des arêtes de $M$ qui ont deux ailes (celles-ci vont être utilisées dans les chemins augmentants).

Une itération de cette procédure est illustrée dans la figure 8. Cette procédure est utilisée pour trouver des chemins augmentants qui sont utilisés pour augmenter la taille du couplage $M$. On répète ce procédé de l'ordre de $\Theta(\log 1/\epsilon)$ fois, en choisissant $\delta = \Theta(\epsilon)$ et on se retrouve avec un couplage dont la taille est $\frac{2}{3} - \epsilon$ plus grande que la taille d'un couplage maximum de $G$.

Pour une étude détaillée de cet algorithme, on se référera à [FKM+05b]. Cet algorithme est généralisé dans [McG05] pour les graphes généraux et pour trouver des chemins augmentants

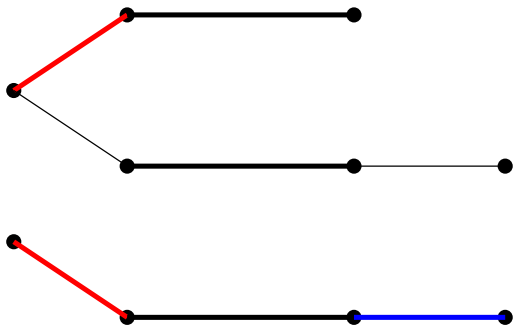

**Fig. 8** – *Procédure pour trouver un ensemble de chemins augmentants de longueur* 3

de n'importe quelle taille, il est probabiliste et trouve un couplage avec un facteur d'approximation $1-\epsilon$ en $O((1/\epsilon)^{(1/\epsilon)}\text{poly}(1/\epsilon))$ passes.