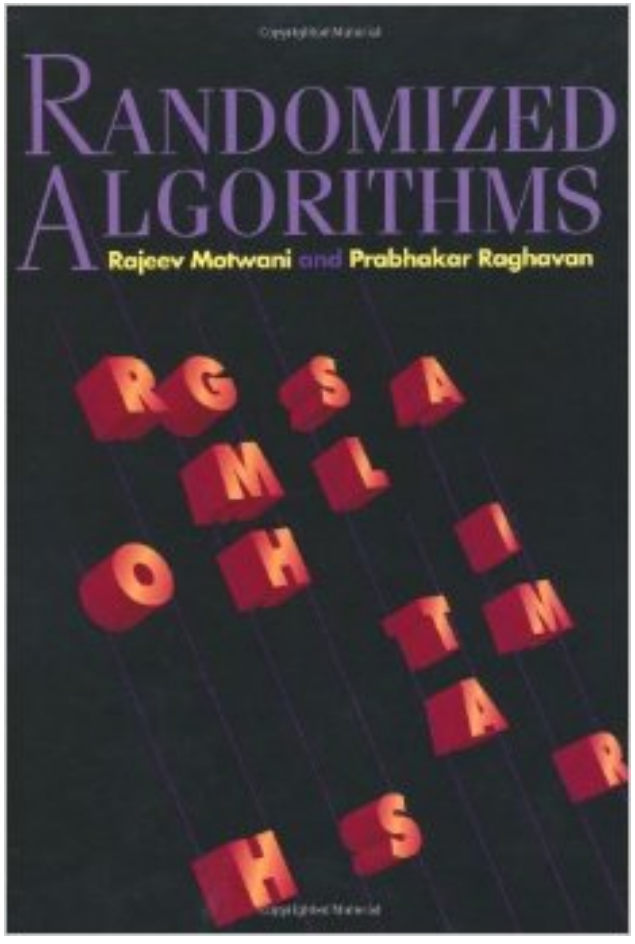
Introduction aux algorithmes randomisés

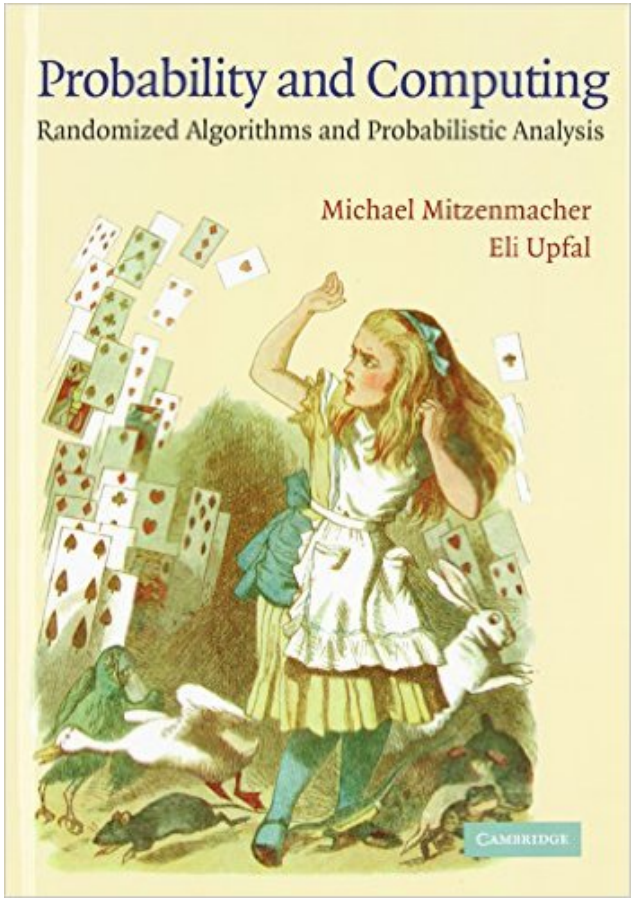
VLADY RAVELOMANANA

IRIF – UMR CNRS 8243
Université de Paris 7
vlad@irif.fr

– M1 Algorithmique avancée –

Bibliographie





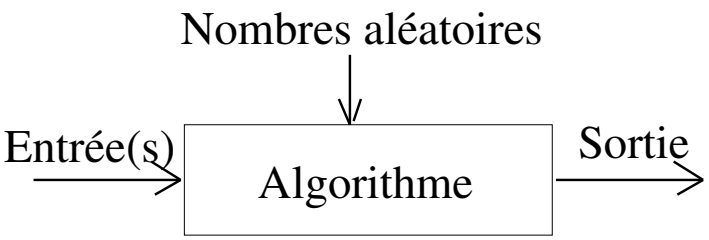
Plan du cours

- Algorithmes randomizés.
  - Motivations des algorithmes randomizés.
  - Calcul du Min-Cut d'un graphe.
  - Le collectionneur de coupons.
  - Problèmes difficiles et algo d'approximation.
  - La méthode probabiliste.

Algorithmes déterministes – randomizés

- Un **algorithme déterministe** a pour but de résoudre un problème donné à partir d'une entrée donnée:

  - Objectifs: l'algorithme doit être correct (toujours résoudre le problème qui lui est assigné) et doit s'éxecuter le plus rapidement possible (par exemple: le nombre d'étapes doit être polynomial en fonction de la taille de l'entrée).
- Un **algorithme randomizé** a les mêmes objectifs qu'un algo déterministe mais il utilise en plus des entrées une source d'aléa, son schéma d'éxecution peut varier sur les mêmes données en entrée:



Forum sur l'algorithme HyperLogLog

PocketTiger

07/10/2014 at 15:05

Excusez moi mais, qu'es que pu[redacted]oO
C'est quoi un "journal probabiliste" et comment ça le journal "répond avec une marge d'erreur" ?
L'informatique c'est le domaine du déterminisme et de l'exactitude par excellence.
Alors il marche comment ce module pifométrique ?
Question subsidiaire, ou est passé le trombone ='(

Cordialement

Anne O'Nyme

07/10/2014 at 15:30

@PocketTiger
Lis le principe de l'algorithme (mal documenté sur wiki certes), il utilise une fonction de hachage appliquée aux entr
franchement pas compliqué. Pour en savoir plus : http://research.neustar.biz/2012/10/25/sketch-of-the-day-hyperlo
data-infrastructure/

Anne O. Nyme

07/10/2014 at 15:46

L'idée c'est d'estimer une approximation. Par exemple, on te donne plein de nombres, tu regarde leur max, et tu re
binaire hein). Forcément, si ton entier le plus grand est du genre 11111111 (255 en décimal) bah tu ne peux pas a
est du genre 00001011 (11 en décimal) bah tu ne peux avoir que les nombres entre 11 et 255, soit 244 nombres. C
observables qui sont faciles à calculer (à peu de choses près, vu qu'on n'a pas forcément besoin du plus petit ou pl
sup ou inf) et on en déduit un comptage approximatif du nombre d'éléments.

Algorithmes randomizés: Las-Vegas et Monte Carlo

Les algorithmes randomizés peuvent être classés en 2 catégories:

- Les algorithmes **`Las-Vegas`** sont des algorithmes randomizés qui produisent toujours **`les mêmes solutions`** que les algorithmes **`déterministes`** . Seuls leur temps d'éxecution sont des variables aléatoires.

- Les algorithmes **`Monte-Carlo`** sont des algorithmes randomizés qui produisent une solution en un temps prévisible (généralement court) avec une certaine probabilité (souvent très proche de 1) qui quantifie le fait que la solution est correcte ou non.

Exemples.

Un grand nombre (N) de personnes (processeurs) anonymes désirent élire un leader selon le modèle suivant:
chaque jour, une (ou plusieurs) personne(s) lève(nt) la main si elle est seule, elle est élue et l'algorithme s'arrête.
Remarque: il n'existe pas de solution déterministe au problème (anonymes!).

Exemples.

Un grand nombre (N) de personnes (processeurs) anonymes désirent élire un leader selon le modèle suivant:
chaque jour, une (ou plusieurs) personne(s) lève(nt) la main si elle est seule, elle est élue et l'algorithme s'arrête.
Remarque: il n'existe pas de solution déterministe au problème (anonymes!).

Algorithme Election-Las-Vegas

while *Pas d'élu* **do**
  Avec probabilité $\frac{1}{N}$ lever la main;
  if *Seule* **then**
    la personne qui a levé la main est élue;
  end
end

Algorithme Election-Monte-Carlo

compteur := 0
while *compteur* $< \log N$ **do**
  Avec probabilité $\frac{1}{N}$ lever la main;
  if *Seule* **then**
    la personne qui a levé la main est élue;
    exit;
  end
  compteur := compteur +1;
end

Exemples.

Un grand nombre (N) de personnes (processeurs) anonymes désirent élire un leader selon le modèle suivant:
chaque jour, une (ou plusieurs) personne(s) lève(nt) la main si elle est seule, elle est élue et l'algorithme s'arrête.
Remarque: il n'existe pas de solution déterministe au problème (anonymes!).

Algorithme Election-Las-Vegas

while *Pas d'élu* **do**
  Avec probabilité $\frac{1}{N}$ lever la main;
  if *Seule* **then**
    la personne qui a levé la main est élue;
  end
end

Algorithme Election-Monte-Carlo

compteur $:= 0$
while *compteur* $< \log N$ **do**
  Avec probabilité $\frac{1}{N}$ lever la main;
  if *Seule* **then**
    la personne qui a levé la main est élue;
    exit;
  end
  compteur $:=$ compteur $+1$;
end

Questions?

Si convergence, convergence en combien de jours? Sinon, quel est le taux d'erreur?

Analyses

Proposition.

L'algorithme Election-Las-Vegas se termine toujours (`avec probabilité 1`) en un temps $\sim$ `Géométrique(` $(1-1/N)^{N-1}$ `)` jours.

Analyses

Proposition.

L'algorithme Election-Las-Vegas se termine toujours (**avec probabilité** 1) en un temps $\sim$ **Géométrique(** $(1-1/N)^{N-1}$ **)** jours.

Preuve. La probabilité qu'il y ait une élection un jour donné est

$$p = \binom{N}{1} \times \frac{1}{N}\left(1-\frac{1}{N}\right)^{N-1} = \left(1-\frac{1}{N}\right)^{N-1}.$$

L'élection a lieu au jour j ($j \geq 1$) avec probabilité: $\underbrace{(1-p)^{j-1}}_{(j-1)\text{ râtés}} \times p$.

Analyses

Proposition.

L'algorithme Election-Las-Vegas se termine toujours (**avec probabilité** 1) en un temps $\sim$ **Géométrique(** $(1-1/N)^{N-1}$**)** jours.

Preuve. La probabilité qu'il y ait une élection un jour donné est

$$p = \binom{N}{1} \times \frac{1}{N}\left(1-\frac{1}{N}\right)^{N-1} = \left(1-\frac{1}{N}\right)^{N-1}.$$

L'élection a lieu au jour j ($j \geq 1$) avec probabilité: $\underbrace{(1-p)^{j-1}}_{(j-1)\text{ râtés}} \times p$.

Proposition.

L'algorithme Election-Monte-Carlo se termine toujours en temps au plus $\log N$ et réussit à élire un leader **avec une très grande probabilité quand** N **est grand** (**au moins** $1-\frac{1}{N^c}$ **pour une constante absolue** $c > 0$).

Analyses

Proposition.

L'algorithme Election-Las-Vegas se termine toujours (**avec probabilité** 1) en un temps $\sim$ **Géométrique(** $(1-1/N)^{N-1}$**)** jours.

Preuve. La probabilité qu'il y ait une élection un jour donné est

$$p = \binom{N}{1} \times \frac{1}{N}\left(1-\frac{1}{N}\right)^{N-1} = \left(1-\frac{1}{N}\right)^{N-1}.$$

L'élection a lieu au jour j ($j \geq 1$) avec probabilité: $\underbrace{(1-p)^{j-1}}_{(j-1)\text{ râtés}} \times p$.

Proposition.

L'algorithme Election-Monte-Carlo se termine toujours en temps au plus $\log N$ et réussit à élire un leader **avec une très grande probabilité quand** N **est grand** (**au moins** $1-\frac{1}{N^c}$ **pour une constante absolue** $c>0$).

Preuve. La boucle est bornée par $\log N$. La probabilité qu'il n'y ait pas d'élu à chaque tour est $1-\frac{1}{e}$ et donc la probabilité d'échec est $(1-e^{-1})^{\log N}$ (échec pour tous les $\log N$ tours).

Initialisation des processeurs en distribué

Le modèle: RADIO NETWORK with CD

- Modèle: processeurs synchrones (temps discret) avec détecteur de collision.

Initialisation des processeurs en distribué

Le modèle: RADIO NETWORK with CD

- Modèle: processeurs synchrones (temps discret) avec détecteur de collision.
- A chaque instant $t \in \mathbb{N}$, un processeur peut envoyer un message sur l'**unique canal** de communication (partagé entre tous les processeurs).

Initialisation des processeurs en distribué

Le modèle: RADIO NETWORK with CD

- Modèle: processeurs synchrones (temps discret) avec détecteur de collision.
- A chaque instant $t \in \mathbb{N}$, un processeur peut envoyer un message sur l'**unique canal** de communication (partagé entre tous les processeurs).
- Un message MSG est **correctement reçu instantannément** `si et seulement si` un **unique processeur** a déposé MSG sur le canal. Le statut du canal est alors dénoté `SINGLE`.

Initialisation des processeurs en distribué

Le modèle: RADIO NETWORK with CD

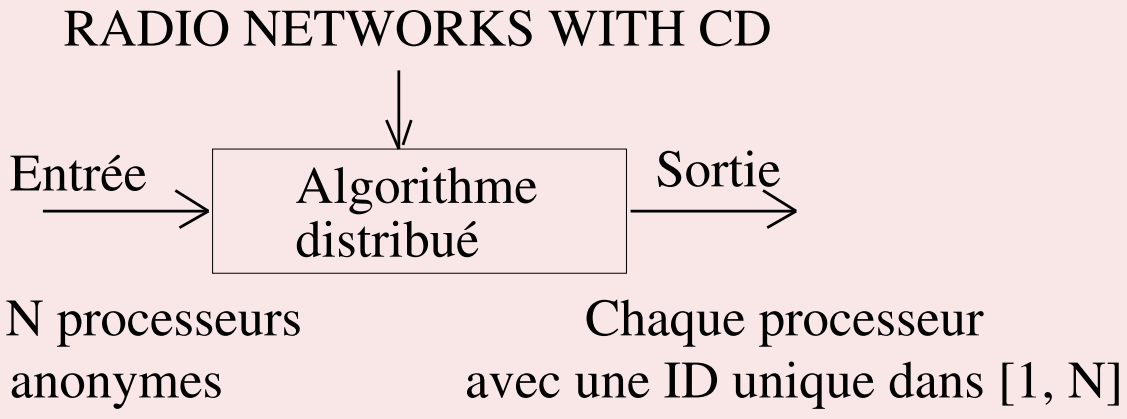
- Modèle: processeurs synchrones (temps discret) avec détecteur de collision.
- A chaque instant $t \in \mathbb{N}$, un processeur peut envoyer un message sur l'**unique canal** de communication (partagé entre tous les processeurs).
- Un message MSG est **correctement reçu instantannément** `si et seulement si` un **unique processeur** a déposé MSG sur le canal. Le statut du canal est alors dénoté `SINGLE`.
- Dans le cas contraire, le statut du canal est soit `NULL` soit `COLLISION`.

Initialisation des processeurs en distribué

Le modèle: RADIO NETWORK with CD

- Modèle: processeurs synchrones (temps discret) avec détecteur de collision.
- A chaque instant $t \in \mathbb{N}$, un processeur peut envoyer un message sur l'**unique canal** de communication (partagé entre tous les processeurs).
- Un message MSG est **correctement reçu instantannément** `si et seulement si` un **unique processeur** a déposé MSG sur le canal. Le statut du canal est alors dénoté `SINGLE`.
- Dans le cas contraire, le statut du canal est soit `NULL` soit `COLLISION`.

Le problème de l'initialisation (namming, DHCP, ...)



L'algorithme distribué de partitionnement d'un ensemble

Protocole `Partition-with-CD`
Répéter {
    chaque station choisit 0 ou 1 avec proba. $\frac{1}{2}$;
    toutes les stations qui ont choisi 1 diffuse au **temps impair**;
    soit `Statut(1)` le statut du canal à ce moment;
    toutes les stations qui ont choisi 0 diffuse au **temps pair**;
    soit `Statut(0)` le statut du canal à ce moment;
} **Jusqu'à** (`Statut(1)` $!=$ `NULL` **et** `Statut(0)` $!=$ `NULL`);

L'algorithme distribué d'initialisation

C'est le prototype même du "**protocole en arbre**". Formellement, l'algorithme utilise la procédure de partitionnement précédente:

Protocole `Initialisation`

Pour toutes les stations faire: **cur** $:= 1$, $L := 1$ et $N := 1$;

while $L \geq 1$ **do**

  if $|P_L| == 1$ **then**

    (

  end

  $\star$ **SINGLE** $\star$) L'unique station **est numérotée N**;

  Toutes les autres stations effectuent $L := L - 1$ et $N := N + 1$;

  else

    Utiliser le module **Partition-with-CD** pour partitionner P_L;

    ($\star$ on aura alors 2 partitions non-vides P_L et P_{L+1} $\star$);

    Toutes les stations effectuent $L := L + 1$;

    Toutes les stations dans P_L effectuent **cur** $:= L$;

  end

end

Analyse du protocole d'initialisation

Binomiales et bornes de Chernoff

$X = \mathrm{Bin}(n, p)$, OU ENCORE $\mathbb{P}[X = r] = \binom{n}{r} p^r (1-p)^{n-r}$. On a

$\mathbb{P}[X \leq (1-\varepsilon)\mathbb{E}[X]] \leq e^{-\frac{\varepsilon^2}{2}\mathbb{E}[X]}$ et $\mathbb{P}[X \leq (1+\varepsilon)\mathbb{E}[X]] \leq e^{-\frac{\varepsilon^3}{2}\mathbb{E}[X]}$, $0 < \varepsilon < 1$

Analyse du protocole d'initialisation

Binomiales et bornes de Chernoff

$X = \mathrm{Bin}(n, p)$, OU ENCORE $\mathbb{P}[X = r] = \binom{n}{r} p^r (1-p)^{n-r}$. On a

$\mathbb{P}[X \leq (1-\varepsilon)\mathbb{E}[X]] \leq e^{-\frac{\varepsilon^2}{2}\mathbb{E}[X]}$ et $\mathbb{P}[X \leq (1+\varepsilon)\mathbb{E}[X]] \leq e^{-\frac{\varepsilon^3}{2}\mathbb{E}[X]}$, $0 < \varepsilon < 1$

Proposition.

Un réseau de N stations opérant sous le modèle des RADIO NETWORKS avec détecteur de collision peut-être initialisé en $O(n)$ étapes avec une probabilité plus grande que $1 - \frac{1}{2^N}$.

Analyse du protocole d'initialisation

Binomiales et bornes de Chernoff

$X = \mathrm{Bin}(n, p)$, OU ENCORE $\mathbb{P}[X = r] = \binom{n}{r} p^r (1-p)^{n-r}$. On a

$\mathbb{P}[X \leq (1-\varepsilon)\mathbb{E}[X]] \leq e^{-\frac{\varepsilon^2}{2}\mathbb{E}[X]}$ et $\mathbb{P}[X \leq (1+\varepsilon)\mathbb{E}[X]] \leq e^{-\frac{\varepsilon^3}{2}\mathbb{E}[X]}$, $0 < \varepsilon < 1$

Proposition.

Un réseau de N stations opérant sous le modèle des RADIO NETWORKS avec détecteur de collision peut-être initialisé en $O(n)$ étapes avec une probabilité plus grande que $1 - \frac{1}{2^N}$.

Preuve.

Au total, le test "$|P_L| =?1$" est donc appelé $2N - 1 = N + N - 1$ fois. On va dire qu'un appel à **`Partition-with-CD`** est réussi s'il partitionne directement l'ensemble en 2 sous-ensembles non vides chacun. Si on fait $8N$ tentatives de partitionnements avec une probabilité d'au moins $\frac{1}{2}$ à chaque fois de réussir directement à partitionner, en utilisant la seconde borne de Chernov avec **`moyenne`** $= 4N$, $\varepsilon = \frac{3}{4}$ on trouve

$$\mathbb{P}[\text{sur 8N tentatives il y a moins que N-1 partitions réussies}] < \frac{1}{2^N}$$

.

Plan du cours

- Algorithmes randomizés.
- Motivations des algorithmes randomizés.
- Calcul du Min-Cut d'un graphe.
- Le collectionneur de coupons.
- Problèmes difficiles et algo d'approximation.
- La méthode probabiliste.

Pourquoi des algorithmes randomizés?

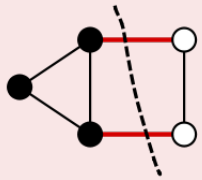
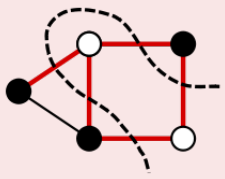
Raison(s)	**Exemple(s)**
Impossibilité de déterminisme	Eléction anonyme, initialisation ✏ algo. distribuée et/ou parallèle
Simplicité	✏ Quicksort ✏ Théorie des nombres (test de primalité, ...)
Rapidité	Identités algébriques: ✏ un polynôme est-il nul $P(x, y) \equiv 0$? ✏ multiplication de matrices $A \times B = C$?
Contraintes de temps de réponse (online, streaming ...)	La décision doit être immédiate et ne dépendra donc que d'une entrée tronquée.
Solutions approchées à des problèmes très complexes	La solution exacte coûte un temps exponentiel (ex: $O(2^n)$) une solution approchée avec une certaine garantie un temps polynomial (ex: $O(n^2)$).
Dérandomization	Ecrire un algo. randomizé et le rendre déterministe en dérandomizant.

Plan du cours

- Algorithmes randomizés.
- Motivations des algorithmes randomizés.
- Calcul du Min-Cut d'un graphe.
- Le collectionneur de coupons.
- Problèmes difficiles et algo d'approximation.
- La méthode probabiliste.

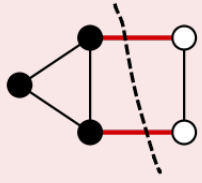
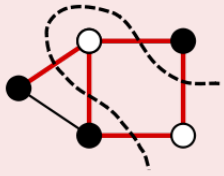
Coupes dans les graphes

Définition: coupe, coupe minimum (MIN-CUT) et coupe maximum (MAX-CUT)

- Une **coupe** d'un graphe est une partition des sommets en deux sous-ensembles.  La **taille** d'une coupe est le nombre d'arêtes entre les deux sous-ensembles.
- Une **coupe maximum** ou **MAX-CUT** est une coupe contenant au moins autant d'arêtes que n'importe quelle autre coupe. 
- Une **coupe minimum** ou **MIN-CUT** est une coupe contenant le plus petit nombre d'arêtes qu'il faut supprimer pour déconnecter le graphe.

Coupes dans les graphes

Définition: coupe, coupe minimum (MIN-CUT) et coupe maximum (MAX-CUT)

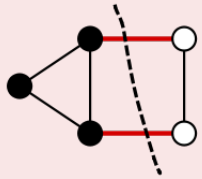
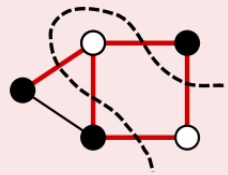
- Une **coupe** d'un graphe est une partition des sommets en deux sous-ensembles.  La **taille** d'une coupe est le nombre d'arêtes entre les deux sous-ensembles.
- Une **coupe maximum** ou **MAX-CUT** est une coupe contenant au moins autant d'arêtes que n'importe quelle autre coupe. 
- Une **coupe minimum** ou **MIN-CUT** est une coupe contenant le plus petit nombre d'arêtes qu'il faut supprimer pour déconnecter le graphe.

Ecrire un **algorithme** pour trouver le MIN-CUT de n'importe quel graphe:

- Entrée: un graphe $G = (V, E)$ (donné sous forme de **listes d'adjacence** ou **matricielle**).
- Sortie: une partition de sommets dans V en $B = \{\textbf{Blancs}\}$ et $N = \{\textbf{Noirs}\}$ telle que le nombre d'arêtes entre N et B soit **minimum** .

Coupes dans les graphes

Définition: coupe, coupe minimum (MIN-CUT) et coupe maximum (MAX-CUT)

- Une **coupe** d'un graphe est une partition des sommets en deux sous-ensembles.  La **taille** d'une coupe est le nombre d'arêtes entre les deux sous-ensembles.
- Une **coupe maximum** ou **MAX-CUT** est une coupe contenant au moins autant d'arêtes que n'importe quelle autre coupe. 
- Une **coupe minimum** ou **MIN-CUT** est une coupe contenant le plus petit nombre d'arêtes qu'il faut supprimer pour déconnecter le graphe.

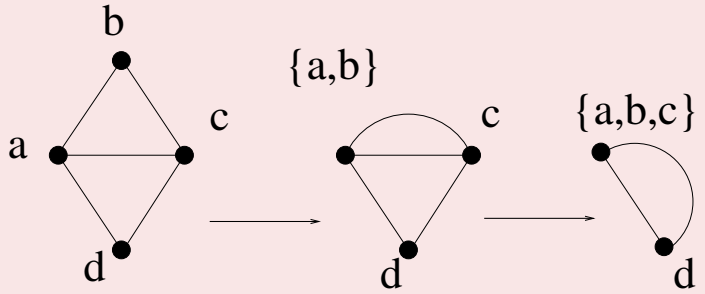
Ecrire un **algorithme** pour trouver le MIN-CUT de n'importe quel graphe:

- Entrée: un graphe $G = (V, E)$ (donné sous forme de **listes d'adjacence** ou **matricielle**).
- Sortie: une partition de sommets dans V en $B = \{\textbf{Blancs}\}$ et $N = \{\textbf{Noirs}\}$ telle que le nombre d'arêtes entre N et B soit **minimum** .

Algorithme déterministe $\longrightarrow$ compliqué!

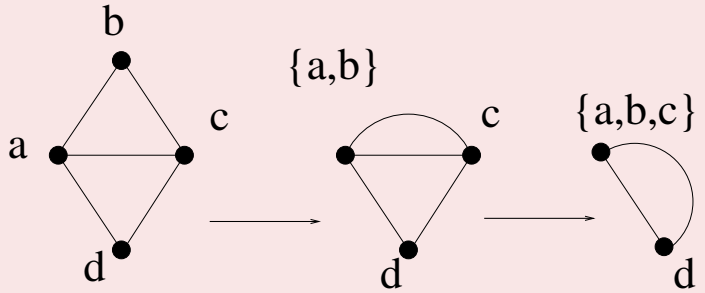
Min-Cut randomizé (Karger 1993)

Définition: contraction d'arêtes d'un graphe



Min-Cut randomizé (Karger 1993)

Définition: contraction d'arêtes d'un graphe



Algorithme Min-Cut de Karger

On définit $S_v = \{v\}$ pour tous les sommets de G;
while $|V| \leq 3$ **do**
    Prendre uniformément au hasard une arête $e = (u, v)$ et la contracter;
    Si z est le nouveau sommet alors $S_z = S_u \bigcup S_v$;
end
Soient u et v les deux derniers sommets, **retourner(**S_u, S_v**)**

Analyse de l'algorithme de Karger

Remarques.

L'algorithme est un **Monte Carlo** : il ne retournera pas toujours le MIN-CUT.
Les questions qui se posent sont alors:

- Avec quelle **probabilité** , il retournera le MIN-CUT du graphe en entrée?
- **Quantifier son temps d'exécution** en fonction de la probabilité?

Analyse de l'algorithme de Karger

Remarques.

L'algorithme est un **Monte Carlo** : il ne retournera pas toujours le MIN-CUT.
Les questions qui se posent sont alors:

- Avec quelle **probabilité** , il retournera le MIN-CUT du graphe en entrée?
- **Quantifier son temps d'exécution** en fonction de la probabilité?

Proposition.

Soit $G = (V, E)$ un graphe avec n sommets et $\mu = (S, \bar{S})$ un MIN-CUT de G.
L'algorithme de Karger trouve μ avec une probabilité supérieure à $\frac{2}{n(n-1)}$.

Analyse de l'algorithme de Karger

Remarques.

L'algorithme est un **Monte Carlo** : il ne retournera pas toujours le MIN-CUT.
Les questions qui se posent sont alors:

- Avec quelle **probabilité** , il retournera le MIN-CUT du graphe en entrée?
- **Quantifier son temps d'exécution** en fonction de la probabilité?

Proposition.

Soit $G = (V, E)$ un graphe avec n sommets et $\mu = (S, \bar{S})$ un MIN-CUT de G.
L'algorithme de Karger trouve μ avec une probabilité supérieure à $\frac{2}{n(n-1)}$.

Amplification.

En répétant l'algorithme $O(n^2 \log n)$ fois, avec une très grande probabilité on trouve un MIN-CUT (**au moins une probabilité de** $1 - \frac{1}{N^c}$ **pour une constante absolue** $c > 0$).

Plan du cours

- Algorithmes randomizés.
- Motivations des algorithmes randomizés.
- Calcul du Min-Cut d'un graphe.
- Le collectionneur de coupons.
- Problèmes difficiles et algo d'approximation.
- La méthode probabiliste.

Le collectionneur de coupons

Le problème.

Dans une collection, nous avons N coupons distincts. A chaque instant t, un collectionneur achète un coupon au hasard.

QUESTION: Au bout de combien de temps, notre collectionneur possèdera tous les N coupons distincts?

Analyse.

Soit X le nombre de coupons achetés jusqu'à ce que le collectionneur possède chacun des N coupons différents.

Au moment où le collectionneur a déjà j types de coupons alors la probabilité d'avoir un nouveau type différent est

Le collectionneur de coupons

Le problème.

Dans une collection, nous avons N coupons distincts. A chaque instant t, un collectionneur achète un coupon au hasard.

QUESTION: Au bout de combien de temps, notre collectionneur possèdera tous les N coupons distincts?

Analyse.

Soit X le nombre de coupons achetés jusqu'à ce que le collectionneur possède chacun des N coupons différents.

Au moment où le collectionneur a déjà j types de coupons alors la probabilité d'avoir un nouveau type différent est $\frac{(N-j)}{N}$

Le collectionneur de coupons

Le problème.

Dans une collection, nous avons N coupons distincts. A chaque instant t, un collectionneur achète un coupon au hasard.

QUESTION: Au bout de combien de temps, notre collectionneur possèdera tous les N coupons distincts?

Analyse.

Soit X le nombre de coupons achetés jusqu'à ce que le collectionneur possède chacun des N coupons différents.

Au moment où le collectionneur a déjà j types de coupons alors la probabilité d'avoir un nouveau type différent est $\frac{(N-j)}{N}$ car sur les N types de coupons, seuls $N-j$ lui permettent de progresser (avoir un nouveau type).

Le collectionneur de coupons

Le problème.

Dans une collection, nous avons N coupons distincts. A chaque instant t, un collectionneur achète un coupon au hasard.
QUESTION: Au bout de combien de temps, notre collectionneur possèdera tous les N coupons distincts?

Analyse.

Soit X le nombre de coupons achetés jusqu'à ce que le collectionneur possède chacun des N coupons différents.
Au moment où le collectionneur a déjà j types de coupons alors la probabilité d'avoir un nouveau type différent est $\frac{(N-j)}{N}$ car sur les N types de coupons, seuls $N-j$ lui permettent de progresser (avoir un nouveau type). On va **diviser** la variable aléatoire X en phases: soit X_j le nombre d'étapes qu'il faut pour avoir **un nouveau type de coupons** quand le collectionneur a déjà **j** coupons. On a

$$X = X_0 + X_1 + \cdots + X_{n-1}$$

qu'on peut lire comme "$j^{\text{ième}}$ **phase se termine quand la** $(j+1)^{\text{ième}}$ **commence**".

Le collectionneur de coupons

Le problème.

Dans une collection, nous avons N coupons distincts. A chaque instant t, un collectionneur achète un coupon au hasard.
QUESTION: Au bout de combien de temps, notre collectionneur possèdera tous les N coupons distincts?

Analyse.

Soit X le nombre de coupons achetés jusqu'à ce que le collectionneur possède chacun des N coupons différents.
Au moment où le collectionneur a déjà j types de coupons alors la probabilité d'avoir un nouveau type différent est $\frac{(N-j)}{N}$ car sur les N types de coupons, seuls $N-j$ lui permettent de progresser (avoir un nouveau type). On va **diviser** la variable aléatoire X en phases: soit X_j le nombre d'étapes qu'il faut pour avoir **un nouveau type de coupons** quand le collectionneur a déjà **j** coupons. On a

$$X = X_0 + X_1 + \cdots + X_{n-1}$$

qu'on peut lire comme "$j^{\text{ième}}$ **phase se termine quand la** $(j+1)^{\text{ième}}$ **commence**". Comme la probabilité d'avoir un nouveau coupon est $\frac{(N-j)}{N}$ l'espérance est

Le collectionneur de coupons

Le problème.

Dans une collection, nous avons N coupons distincts. A chaque instant t, un collectionneur achète un coupon au hasard.
QUESTION: Au bout de combien de temps, notre collectionneur possèdera tous les N coupons distincts?

Analyse.

Soit X le nombre de coupons achetés jusqu'à ce que le collectionneur possède chacun des N coupons différents.
Au moment où le collectionneur a déjà j types de coupons alors la probabilité d'avoir un nouveau type différent est $\frac{(N-j)}{N}$ car sur les N types de coupons, seuls $N-j$ lui permettent de progresser (avoir un nouveau type). On va **diviser** la variable aléatoire X en phases: soit X_j le nombre d'étapes qu'il faut pour avoir **un nouveau type de coupons** quand le collectionneur a déjà **j** coupons. On a

$$X = X_0 + X_1 + \cdots + X_{n-1}$$

qu'on peut lire comme "$j^{\text{ième}}$ **phase se termine quand la** $(j+1)^{\text{ième}}$ **commence**". Comme la probabilité d'avoir un nouveau coupon est $\frac{(N-j)}{N}$ l'espérance est $\mathbb{E}[X_j] = \frac{N}{N-j}$.

Le collectionneur de coupons

Le problème.

Dans une collection, nous avons N coupons distincts. A chaque instant t, un collectionneur achète un coupon au hasard.
QUESTION: Au bout de combien de temps, notre collectionneur possèdera tous les N coupons distincts?

Analyse.

Soit X le nombre de coupons achetés jusqu'à ce que le collectionneur possède chacun des N coupons différents.
Au moment où le collectionneur a déjà j types de coupons alors la probabilité d'avoir un nouveau type différent est $\frac{(N-j)}{N}$ car sur les N types de coupons, seuls $N-j$ lui permettent de progresser (avoir un nouveau type). On va **diviser** la variable aléatoire X en phases: soit X_j le nombre d'étapes qu'il faut pour avoir **un nouveau type de coupons** quand le collectionneur a déjà **j** coupons. On a

$$X = X_0 + X_1 + \cdots + X_{n-1}$$

qu'on peut lire comme "$j^{\text{ième}}$ **phase se termine quand la** $(j+1)^{\text{ième}}$ **commence**". Comme la probabilité d'avoir un nouveau coupon est $\frac{(N-j)}{N}$ l'espérance est $\mathbb{E}[X_j] = \frac{N}{N-j}$. En utilisant la linéarité de l'espérance on montre alors que

Le collectionneur de coupons

Le problème.

Dans une collection, nous avons N coupons distincts. A chaque instant t, un collectionneur achète un coupon au hasard.
QUESTION: Au bout de combien de temps, notre collectionneur possèdera tous les N coupons distincts?

Analyse.

Soit X le nombre de coupons achetés jusqu'à ce que le collectionneur possède chacun des N coupons différents.
Au moment où le collectionneur a déjà j types de coupons alors la probabilité d'avoir un nouveau type différent est $\frac{(N-j)}{N}$ car sur les N types de coupons, seuls $N-j$ lui permettent de progresser (avoir un nouveau type). On va **diviser** la variable aléatoire X en phases: soit X_j le nombre d'étapes qu'il faut pour avoir **un nouveau type de coupons** quand le collectionneur a déjà **j** coupons. On a

$$X = X_0 + X_1 + \cdots + X_{n-1}$$

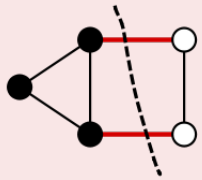
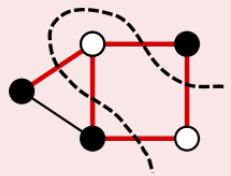
qu'on peut lire comme "$j^{\text{ième}}$ **phase se termine quand la** $(j+1)^{\text{ième}}$ **commence**". Comme la probabilité d'avoir un nouveau coupon est $\frac{(N-j)}{N}$ l'espérance est $\mathbb{E}[X_j] = \frac{N}{N-j}$. En utilisant la linéarité de l'espérance on montre alors que $\mathbb{E}[X] = \Theta(N \log N)$.

Plan du cours

- Algorithmes randomizés.
- Motivations des algorithmes randomizés.
- Calcul du Min-Cut d'un graphe.
- Le collectionneur de coupons.
- Problèmes difficiles et algo d'approximation.
- La méthode probabiliste.

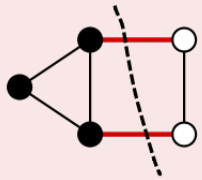
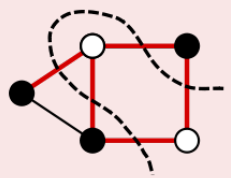
Un exemple de problème difficile: MAX-CUT

Définition: MAX-CUT ou coupe maximum

- Une **coupe** d'un graphe est une partition des sommets en deux sous-ensembles.  La **taille** d'une coupe est le nombre d'arêtes entre les deux sous-ensembles.
- Une **coupe maximum** (couramment appelé MAX-CUT) est une coupe contenant au moins autant d'arêtes que n'importe quelle autre coupe. 

Un exemple de problème difficile: MAX-CUT

Définition: MAX-CUT ou coupe maximum

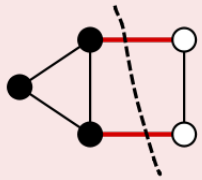
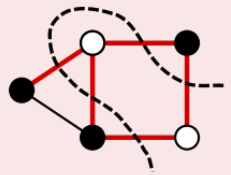
- Une **coupe** d'un graphe est une partition des sommets en deux sous-ensembles.  La **taille** d'une coupe est le nombre d'arêtes entre les deux sous-ensembles.
- Une **coupe maximum** (couramment appelé MAX-CUT) est une coupe contenant au moins autant d'arêtes que n'importe quelle autre coupe. 

Ecrire un **algorithme** tel que

- Entrée: un graphe $G = (V, E)$
- Sortie: une partition de sommets dans V en $B = \{\textbf{Blancs}\}$ et $N = \{\textbf{Noirs}\}$ telle que le nombre d'arêtes entre N et B soit **maximum**.

Un exemple de problème difficile: MAX-CUT

Définition: MAX-CUT ou coupe maximum

- Une **coupe** d'un graphe est une partition des sommets en deux sous-ensembles.  La **taille** d'une coupe est le nombre d'arêtes entre les deux sous-ensembles.
- Une **coupe maximum** (couramment appelé MAX-CUT) est une coupe contenant au moins autant d'arêtes que n'importe quelle autre coupe. 

Ecrire un **algorithme** tel que

- Entrée: un graphe $G = (V, E)$
- Sortie: une partition de sommets dans V en $B = \{\textbf{Blancs}\}$ et $N = \{\textbf{Noirs}\}$ telle que le nombre d'arêtes entre N et B soit **maximum**.

Algorithme déterministe.

Une idée (gloutonne) serait de regarder toutes les possibilités: $O(2^n)$. Les seuls algorithmes exacts connus s'éxècutent en temps exponentiel.

Solution approchée avec garanti pour MAX-CUT

Algorithme d'approximation

Un **algorithme d'approximation** est un algorithme permettant de calculer une **solution approchée** à un problème algorithmique difficile.

Solution approchée avec garanti pour MAX-CUT

Algorithme d'approximation

Un **algorithme d'approximation** est un algorithme permettant de calculer une **solution approchée** à un problème algorithmique difficile.

Concrètement.

On doit trouver une solution **maximum** (resp. **minimum**) à un problème donné. On écrit un algorithme efficace en temps et en mémoire qui fournit une solution telle que celle-ci est **au moins la moitié du maximum** (resp. au plus la moitié du minimum) **sur toutes les instances** du problème (sachant que trouver le maximum/minimum nécessitera un temps déraisonnable).

Solution approchée avec garanti pour MAX-CUT

Algorithme d'approximation

Un **algorithme d'approximation** est un algorithme permettant de calculer une **solution approchée** à un problème algorithmique difficile.

Concrètement.

On doit trouver une solution **maximum** (resp. **minimum**) à un problème donné. On écrit un algorithme efficace en temps et en mémoire qui fournit une solution telle que celle-ci est **au moins la moitié du maximum** (resp. au plus la moitié du minimum) **sur toutes les instances** du problème (sachant que trouver le maximum/minimum nécessitera un temps déraisonnable).

Le glouton.

$G = (V, E)$, on va partitionner V en A et B de la manière gloutonne suivante:
Avec probabilité $1/2$, chaque sommet de V est affecté à A.

Solution approchée avec garanti pour MAX-CUT

Algorithme d'approximation

Un **algorithme d'approximation** est un algorithme permettant de calculer une **solution approchée** à un problème algorithmique difficile.

Concrètement.

On doit trouver une solution **maximum** (resp. **minimum**) à un problème donné. On écrit un algorithme efficace en temps et en mémoire qui fournit une solution telle que celle-ci est **au moins la moitié du maximum** (resp. au plus la moitié du minimum) **sur toutes les instances** du problème (sachant que trouver le maximum/minimum nécessitera un temps déraisonnable).

Le glouton.

$G = (V, E)$, on va partitionner V en A et B de la manière gloutonne suivante:
Avec probabilité $1/2$, chaque sommet de V est affecté à A.

Proposition.

Le glouton retourne une solution $\mathcal{S}$ telle que : $\mathbb{E}[\mathcal{S}] \geq \frac{||\text{solution optimale}||}{2}$.

Un autre exemple de problème difficile: MAX-2-SAT

Définition de 2-SAT.

Soit $X = \{x_1, x_2, \cdots, x_n\}$ un ensemble de variables booléennes. Une clause 2-SAT $\mathcal{C}$ sur X est de la forme $\mathcal{C} = x_u \vee x_v$ avec $(x_u, x_v) \in (X \bigcup \{\neg x_1, \neg x_2, \cdots, \neg x_n\})^2$.

Un autre exemple de problème difficile: MAX-2-SAT

Définition de 2-SAT.

Soit $X = \{x_1, x_2, \cdots, x_n\}$ un ensemble de variables booléennes. Une clause 2-SAT $\mathcal{C}$ sur X est de la forme $\mathcal{C} = x_u \vee x_v$ avec $(x_u, x_v) \in (X \bigcup \{\neg x_1, \neg x_2, \cdots, \neg x_n\})^2$. Une **formule 2-SAT** , notée $\mathcal{F}_{n,M}$ est alors définie par l'ensemble de M clauses $\mathcal{C}_i$, $1 \leq i \leq M$.

Un autre exemple de problème difficile: MAX-2-SAT

Définition de 2-SAT.

Soit $X = \{x_1, x_2, \cdots, x_n\}$ un ensemble de variables booléennes. Une clause 2-SAT $\mathcal{C}$ sur X est de la forme $\mathcal{C} = x_u \vee x_v$ avec $(x_u, x_v) \in (X \bigcup \{\neg x_1, \neg x_2, \cdots, \neg x_n\})^2$. Une **formule 2-SAT** , notée $\mathcal{F}_{n,M}$ est alors définie par l'ensemble de M clauses $\mathcal{C}_i$, $1 \leq i \leq M$.

On dit que la formule $\mathcal{F}_{n,M}$ est **SAT** s'il existe une affectation des variables x_j $(1 \leq j \leq n)$ telle que chacune des clauses $\mathcal{C}_i$ $1 \leq i \leq M$ de $\mathcal{F}_{n,M}$ soit satisfaite.

Dans le cas contraire on dit que la formule $\mathcal{F}_{n,M}$ est **UNSAT**.

Un autre exemple de problème difficile: MAX-2-SAT

Définition de 2-SAT.

Soit $X = \{x_1, x_2, \cdots, x_n\}$ un ensemble de variables booléennes. Une clause 2-SAT $\mathcal{C}$ sur X est de la forme $\mathcal{C} = x_u \vee x_v$ avec $(x_u, x_v) \in (X \bigcup \{\neg x_1, \neg x_2, \cdots, \neg x_n\})^2$. Une **formule 2-SAT** , notée $\mathcal{F}_{n,M}$ est alors définie par l'ensemble de M clauses $\mathcal{C}_i$, $1 \leq i \leq M$.
On dit que la formule $\mathcal{F}_{n,M}$ est **SAT** s'il existe une affectation des variables x_j $(1 \leq j \leq n)$ telle que chacune des clauses $\mathcal{C}_i$ $1 \leq i \leq M$ de $\mathcal{F}_{n,M}$ soit satisfaite.
Dans le cas contraire on dit que la formule $\mathcal{F}_{n,M}$ est **UNSAT**.

Exemples.

Formule n = 3, M = 4 SAT

$$x_1 \vee \neg x_2$$
$$x_3 \vee \neg x_1$$
$$x_2 \vee x_3$$
$$\neg x_2 \vee \neg x_3$$

Formule n = 3, M = 4 UNSAT

$$x_1 \vee \neg x_2$$
$$x_1 \vee x_2$$
$$x_3 \vee \neg x_1$$
$$\neg x_3 \vee \neg x_1$$

2-SAT, graphes dirigés et MAX-2-SAT

Proposition.

Le problème de **décider** si une formule 2-SAT est `SAT` ou `UNSAT` se traite en temps **POLYNOMIAL**.

2-SAT, graphes dirigés et MAX-2-SAT

Proposition.

Le problème de **décider** si une formule 2-SAT est `SAT` ou `UNSAT` se traite en temps **POLYNOMIAL**.

Remarque.

Si la formule est UNSAT, il existe un problème d'**optimisation** consistant à trouver une `sous-formule SAT de taille maximale` : le problème MAX-2-SAT.
MAX-2-SAT est difficile (pas d'algorithme polynomial connu).

Solution approchée avec garanti pour MAX-k-SAT

k-SAT (resp. MAX-k-SAT) généralise 2-SAT (resp. MAX-2-SAT).

Solution approchée avec garanti pour MAX-k-SAT

k-SAT (resp. MAX-k-SAT) généralise 2-SAT (resp. MAX-2-SAT).
L'algorithme consiste à affecter chacune des variables $x_1, x_2, \cdots, x_n$ de manière indépendante à 0 ou à 1 avec probabilité $\frac{1}{2}$.

Solution approchée avec garanti pour MAX-k-SAT

k-SAT (resp. MAX-k-SAT) généralise 2-SAT (resp. MAX-2-SAT).
L'algorithme consiste à affecter chacune des variables $x_1, x_2, \cdots, x_n$ de manière indépendante à 0 ou à 1 avec probabilité $\frac{1}{2}$.
La probabilité de satisfaire une clause C_i quelconque est

$$1 - \frac{1}{2^k}.$$

Soit Z_i la variable aléatoire

$Z_i = 1$ si la clause C_i est satisfaite. Sinon $Z_i = 0$.

Solution approchée avec garanti pour MAX-k-SAT

k-SAT (resp. MAX-k-SAT) généralise 2-SAT (resp. MAX-2-SAT).
L'algorithme consiste à affecter chacune des variables $x_1, x_2, \cdots, x_n$ de manière indépendante à 0 ou à 1 avec probabilité $\frac{1}{2}$.
La probabilité de satisfaire une clause C_i quelconque est

$$1 - \frac{1}{2^k}.$$

Soit Z_i la variable aléatoire

$$Z_i = 1 \text{ si la clause } C_i \text{ est satisfaite. Sinon } Z_i = 0.$$

Par linéarité de l'espérance, le nombre moyen de clauses satisfaites sur M clauses est

$$\mathbb{E}[Z_1 + Z_2 + \cdots + Z_M] = \left(1 - \frac{1}{2^k}\right) M.$$

On a donc le résultat d'approximation suivant:

Solution approchée avec garanti pour MAX-k-SAT

k-SAT (resp. MAX-k-SAT) généralise 2-SAT (resp. MAX-2-SAT).
L'algorithme consiste à affecter chacune des variables $x_1, x_2, \cdots, x_n$ de manière indépendante à 0 ou à 1 avec probabilité $\frac{1}{2}$.
La probabilité de satisfaire une clause C_i quelconque est

$$1 - \frac{1}{2^k}.$$

Soit Z_i la variable aléatoire

$$Z_i = 1 \text{ si la clause } C_i \text{ est satisfaite. Sinon } Z_i = 0.$$

Par linéarité de l'espérance, le nombre moyen de clauses satisfaites sur M clauses est

$$\mathbb{E}[Z_1 + Z_2 + \cdots + Z_M] = \left(1 - \frac{1}{2^k}\right) M.$$

On a donc le résultat d'approximation suivant:

Proposition.

Le nombre moyen de clauses satisfaites par une affectation aléatoire uniforme des variables x_i est une $\frac{2^k-1}{2^k}$ -**approximation** de l'optimale.

Solution approchée avec garanti pour MAX-k-SAT

k-SAT (resp. MAX-k-SAT) généralise 2-SAT (resp. MAX-2-SAT).
L'algorithme consiste à affecter chacune des variables $x_1, x_2, \cdots, x_n$ de manière indépendante à 0 ou à 1 avec probabilité $\frac{1}{2}$.
La probabilité de satisfaire une clause C_i quelconque est

$$1 - \frac{1}{2^k}.$$

Soit Z_i la variable aléatoire

$$Z_i = 1 \text{ si la clause } C_i \text{ est satisfaite. Sinon } Z_i = 0.$$

Par linéarité de l'espérance, le nombre moyen de clauses satisfaites sur M clauses est

$$\mathbb{E}[Z_1 + Z_2 + \cdots + Z_M] = \left(1 - \frac{1}{2^k}\right) M.$$

On a donc le résultat d'approximation suivant:

Proposition.

Le nombre moyen de clauses satisfaites par une affectation aléatoire uniforme des variables x_i est une $\frac{2^k-1}{2^k}$ -**approximation** de l'optimale.

Remarque. Il se trouve par exemple que MAX-3-SAT ne peut-être approximé à un rapport plus grand que $\frac{7}{8}$ à moins que $P = NP$.

Plan du cours

- Algorithmes randomizés.
- Motivations des algorithmes randomizés.
- Calcul du Min-Cut d'un graphe.
- Le collectionneur de coupons.
- Problèmes difficiles et algo d'approximation.
- La méthode probabiliste.

La méthode probabiliste

Simple et puissante

Méthode très puissante basée sur les observations:

- Si $\mathbb{E}[X] \geq \alpha$ alors il existe une instance X telle que $X \geq \alpha$.
- Soit $\mathcal{E}$ un évènement. Si $\mathbb{P}[\mathcal{E}] > 0$ alors il existe une instance telle que $\mathcal{E}$ soit VRAI .

Nombre de Ramsey

Définition à partir des graphes

Le nombre de Ramsey $R(k, h)$ est défini comme étant **le plus petit entier** n tel que dans un graphe à n sommets:

- soit on a un sous-graphe complet de taille k
- soit on a un ensemble indépendant de taille h

Nombre de Ramsey

Définition à partir des graphes

Le nombre de Ramsey $R(k, h)$ est défini comme étant **le plus petit entier** n tel que dans un graphe à n sommets:

- soit on a un sous-graphe complet de taille k
- soit on a un ensemble indépendant de taille h

Exemples

$$R(3, 3) = 6 \text{ et } R(4, 4) = 18\,.$$

Nombre de Ramsey

Définition à partir des graphes

Le nombre de Ramsey $R(k, h)$ est défini comme étant **le plus petit entier** n tel que dans un graphe à n sommets:

- soit on a un sous-graphe complet de taille k
- soit on a un ensemble indépendant de taille h

Exemples

$$R(3, 3) = 6 \text{ et } R(4, 4) = 18\,.$$

Mais on n'a que des bornes après

$$43 \leq R(5, 5) \leq 49\,.$$

Borne de de $R(k, k)$ par Erdős (1947)

Proposition

Si $k \geq 3$ alors $R(k, k) > 2^{\frac{k}{2}}$.

Borne de de $R(k, k)$ par Erdős (1947)

Proposition

Si $k \geq 3$ alors $R(k, k) > 2^{\frac{k}{2}}$.

Preuve. Pour chaque paire de sommets u, v du graphe avec probabilité $\frac{1}{2}$, on décide de mettre une arête. sur $\mathbb{G}\left(n, p = \frac{1}{2}\right)$).

Borne de de $R(k, k)$ par Erdős (1947)

Proposition

Si $k \geq 3$ alors $R(k, k) > 2^{\frac{k}{2}}$.

Preuve. Pour chaque paire de sommets u, v du graphe avec probabilité $\frac{1}{2}$, on décide de mettre une arête. sur $\mathbb{G}\left(n, p = \frac{1}{2}\right)$). La probabilité qu'un ensemble de k sommets soit un ensemble indépendant est $2^{-\binom{k}{2}}$ (de même pour un clique). Donc, la probabilité qu'on ait soit un ensemble indépendant de taille k ou un clique de taille k est $2\frac{\binom{n}{k}}{2^{\binom{k}{2}}}$. Pour $n = 2^{k/2}$, on peut calculer

$$2\frac{\binom{n}{k}}{2^{\binom{k}{2}}} \leq 2\frac{n^k}{k!}2^{-\binom{k}{2}} = \frac{2^{1+k/2}}{k!} < 1\,.$$

(la dernière inégalité étant vraie à partir de 3.)

Borne de de $R(k, k)$ par Erdős (1947)

Proposition

Si $k \geq 3$ alors $R(k, k) > 2^{\frac{k}{2}}$.

Preuve. Pour chaque paire de sommets u, v du graphe avec probabilité $\frac{1}{2}$, on décide de mettre une arête. sur $\mathbb{G}\left(n, p = \frac{1}{2}\right)$). La probabilité qu'un ensemble de k sommets soit un ensemble indépendant est $2^{-\binom{k}{2}}$ (de même pour un clique). Donc, la probabilité qu'on ait soit un ensemble indépendant de taille k ou un clique de taille k est $2\frac{\binom{n}{k}}{2^{\binom{k}{2}}}$. Pour $n = 2^{k/2}$, on peut calculer

$$2\frac{\binom{n}{k}}{2^{\binom{k}{2}}} \leq 2\frac{n^k}{k!}2^{-\binom{k}{2}} = \frac{2^{1+k/2}}{k!} < 1\,.$$

(la dernière inégalité étant vraie à partir de 3.) Comme la probabilité d'avoir un clique de taille k ou un ensemble indépendant de taille k est strictement plus petite que 1, l'évènement contraire a lieu avec probabilité strictement plus grande que 0. Donc, $R(k, k) > 2^{k/2}$

2-colorabilité des hypergraphs k-uniformes

Définitions

- Un hypergraphe k-uniforme H est formé d'un couple $H = (V, E)$ avec $E \subseteq V^k$ (V est l'ensemble des sommets et E celui des hyperarêtes).
- Un hypergraphe est dit 2-colorable si chacun de ses sommets peut être colorer en BLEU et ROUGE de manière à ne jamais avoir d'(hyper)arête monochromatique.
- Soit $m(k)$ le plus petit nombre d'arêtes tel qu'un hypergraphe k uniforme ne soit pas 2-colorable.

2-colorabilité des hypergraphs k-uniformes

Définitions

- Un hypergraphe k-uniforme H est formé d'un couple $H = (V, E)$ avec $E \subseteq V^k$ (V est l'ensemble des sommets et E celui des hyperarêtes).
- Un hypergraphe est dit 2-colorable si chacun de ses sommets peut être colorer en BLEU et ROUGE de manière à ne jamais avoir d'(hyper)arête monochromatique.
- Soit $m(k)$ le plus petit nombre d'arêtes tel qu'un hypergraphe k uniforme ne soit pas 2-colorable.

Remarque. On a $m(2) = 3$ à cause du triangle.

2-colorabilité des hypergraphs k-uniformes

Définitions

- Un hypergraphe k-uniforme H est formé d'un couple $H = (V, E)$ avec $E \subseteq V^k$ (V est l'ensemble des sommets et E celui des hyperarêtes).
- Un hypergraphe est dit 2-colorable si chacun de ses sommets peut être colorer en BLEU et ROUGE de manière à ne jamais avoir d'(hyper)arête monochromatique.
- Soit $m(k)$ le plus petit nombre d'arêtes tel qu'un hypergraphe k uniforme ne soit pas 2-colorable.

Remarque. On a $m(2) = 3$ à cause du triangle.

Proposition

Pour $k \geq 2$ $m(k) \geq 2^{k-1}$.

2-colorabilité des hypergraphs k-uniformes

Définitions

- Un hypergraphe k-uniforme H est formé d'un couple $H = (V, E)$ avec $E \subseteq V^k$ (V est l'ensemble des sommets et E celui des hyperarêtes).
- Un hypergraphe est dit 2-colorable si chacun de ses sommets peut être colorer en BLEU et ROUGE de manière à ne jamais avoir d'(hyper)arête monochromatique.
- Soit $m(k)$ le plus petit nombre d'arêtes tel qu'un hypergraphe k uniforme ne soit pas 2-colorable.

Remarque. On a $m(2) = 3$ à cause du triangle.

Proposition

Pour $k \geq 2$ $m(k) \geq 2^{k-1}$.

Preuve. Soit H tel que $E(H) < 2^{k-1}$. On colorie chaque sommet de H en bleu ou en rouge avec probabilité $\frac{1}{2}$. La probabilité qu'un arête soit monochromatique est donc 2^{1-k} . Si H est tel que $V(H) < 2^{k-1}$, la probabilité qu'il existe une arête monochromatique dans H est (indépendance) strictement plus petite que $2^{k-1}2^{1-k}$. Donc avec une **probabilité strictement non-nulle** H **est** 2**-colorable**.

Ensemble indépendant dans un graphe

Définition

Soit G un graphe. On note (de manière traditionnelle) par $\alpha(G)$ la taille du plus grand ensemble indépendant de G.

Ensemble indépendant dans un graphe

Définition

Soit G un graphe. On note (de manière traditionnelle) par $\alpha(G)$ la taille du plus grand ensemble indépendant de G.

Proposition

Soit $G = (V, E)$ avec $|V| = n$ et $|E| = m$. Soit $d = \frac{2m}{n} \geq 1$ le degré moyen.
Alors

$$\alpha(G) \geq \frac{n}{2d}\,.$$

Preuve. On construit $S \subseteq V$ en insérant dans S chaque sommet avec une probabilité p (qu'on fixera plus tard). On a donc $\mathbb{E}|S| = np$ et si Y est le nombre d'arêtes entre 2 sommets de S on a $\mathbb{E}Y = mp^2 = \frac{ndp^2}{2}$. Mais aussi et surtout

$$\mathbb{E}[X - Y] = np(1 - 1/2dp)\,.$$

Donc il existe $S \subseteq V$ tel que la différence du nombre de sommets et d'arêtes soit au moins $A(p) = np(1 - 1/2dp)$. Maintenant, on **modifie** ce S en enlevant un sommet pour chaque arête dans S. On a un indépendant de taille au moins $A(p)$. On maximise $A(p)$ avec $p = 1/d$.

Nombre chromatique et cycles d'un graphe

Définitions

- Une coloration **propre** d'un graphe G consiste à colorer les sommets de G de telle sorte à ce que deux sommets adjacents soient colorés avec deux couleurs différentes.
- Le **nombre chromatique** d'un graphe G est le nombre minimum de couleurs à utiliser pour le colorer proprement.

Nombre chromatique et cycles d'un graphe

Définitions

- Une coloration **propre** d'un graphe G consiste à colorer les sommets de G de telle sorte à ce que deux sommets adjacents soient colorés avec deux couleurs différentes.
- Le **nombre chromatique** d'un graphe G est le nombre minimum de couleurs à utiliser pour le colorer proprement.

Cycle court et nombre chromatique arbitrairement grands

Pour tout $k, \ell > 0$, il existe un graphe G dont

- le nombre chromatique est strictement plus grand que k et
- le cycle le plus court est strictement plus grand que ℓ.

Remarque. Le résultat est contre-intuitif.

Preuve d'existence (1/2)

On considère les graphes $\mathbb{G}(n, p)$ avec $p = \frac{1}{n^{1-\varepsilon}}$ et $\varepsilon = \frac{1}{2\ell}$.

Preuve d'existence (1/2)

On considère les graphes $\mathbb{G}(n, p)$ avec $p = \frac{1}{n^{1-\varepsilon}}$ et $\varepsilon = \frac{1}{2\ell}$. Soit X la v.a. qui compte le nombre de cycles de longueur **au plus** ℓ.
On calcule

$$\mathbb{E}[X] = \sum_{i=3}^{\ell} \binom{n}{i} \times \frac{1}{2}(i-1)!p^i \,.$$

Comme $\binom{n}{i} \times \frac{1}{2}(i-1)! \leq n^i$, on a

$$\mathbb{E}[X] \leq \sum_{i=3}^{\ell} n^{\varepsilon i} = o(n)$$

. On peut choisir n (on a le choix du graphe!) suffisament large de telle sorte à ce que $\mathbb{E}[X] < \frac{n}{4}$, on a (Markov):

$$\mathbb{P}\left[X \geq \frac{n}{2}\right] < \frac{1}{2}\,.$$

Preuve d'existence (2/2)

Soit $a = \lceil \frac{3}{p} \log n \rceil$. On a ($\alpha(G)$ étant le nombre indépendant de G):

$$\mathbb{P}[\alpha(\mathbb{G}(n,p)) \geq a] \leq \binom{n}{a}(1-p)^{\binom{a}{2}} \leq n^a e^{-p\binom{a}{2}} = e^{a\log n - pa(a-1)/2}$$

qui tend vers 0 quand n **est grand**. Donc pour n suffisament grand $\mathbb{P}[\alpha(\mathbb{G}(n,p)) \geq a] < \frac{1}{2}$. Ainsi, il existe G graphe avec $X < n/2$ et $\alpha(G) < a$. De ce graphe, on enlève un sommet à tous ses cycles, et on obtient un graphe G' avec au moins $n/2$ sommets tel que **le cycle le plus court soit** $> \ell$ et $\alpha(G') < a$. Mais on a

$$\text{nombre chromatique}(G') \geq \frac{n}{2(a-1)} \geq \frac{pn}{6\log n} = \frac{n^\varepsilon}{6\log n}$$

Donc pour n grand **ce nombre chromatique est** $> k$. CQFD