Informatique Quantique

Frédéric Magniez

Cours 5 :
Algorithme de Kitaev, marches quantiques

Problème

- Entrée : un état $\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{i\alpha y}|y\rangle$ pour un angle $0 \le \alpha \le 2\pi$
- Sortie : les n premiers bits de $\alpha/2\pi$

Solution

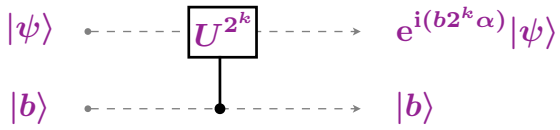
- Cas $\alpha = 2\pi\, x/2^n$: La transformée de Fourier inverse donne x
- Cas général : La transformée de Fourier inverse donne x telle que
  si $n=m+1+\log(1+1/\varepsilon)$, les m premiers bits x' de x satisfont

$$\Pr\left(\left|\frac{x'}{2^m} - \alpha\right| \le \frac{1}{2^{m+1}}\right) \ge 1 - \varepsilon$$

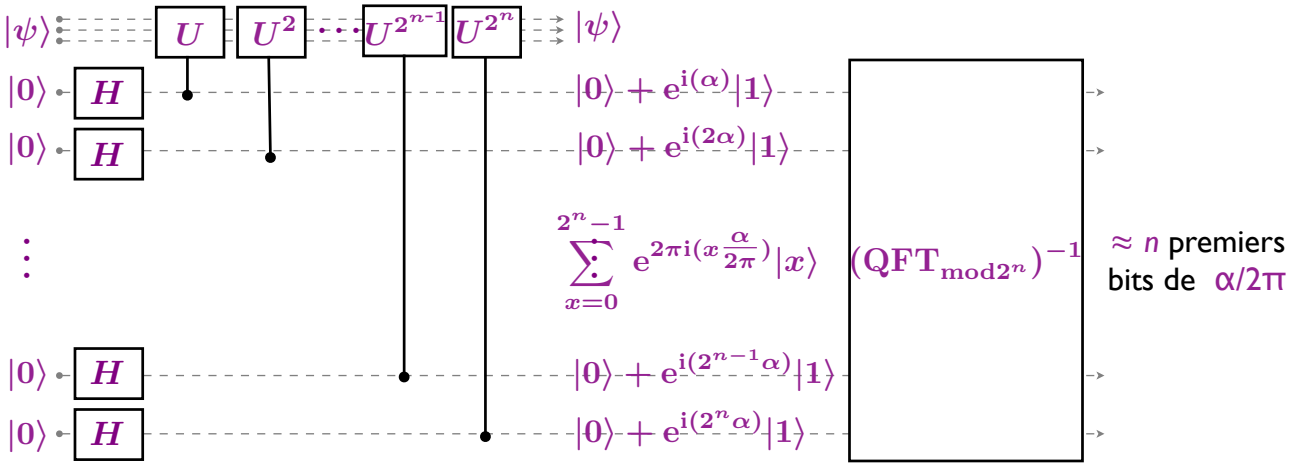
Application : Estimation de valeur propre

- Entrée
  Des boîtes noires réalisant c-U^t pour U unitaire et $t=1,2,\ldots,2^n$
  Un état $|\psi\rangle$ vecteur propre de U pour la valeur propre $e^{i\alpha}$
- Sortie
  La valeur de $\alpha/2\pi$ à n bits de précision près
- Exercice : montrer comment résoudre le problème avec la QFT inverse
  Indication : construire l'état $\frac{1}{\sqrt{2^n}} \sum e^{i\alpha y}|y\rangle$ avec n boîtes c-U^t

Gates



Full circuit



Opérateur Multiplication

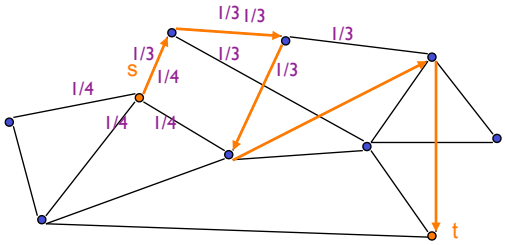
- Soit a un entier modulo N, premier avec N, d'ordre r
- Soit la transformation : $U_a : |x\rangle \mapsto |ax\rangle$
- Montrer que : $(U_a)^t = U_{a^t}$
- En déduire que les valeurs propres λ de U_a satisfont $\lambda^r=1$
- Montrer que chaque vecteur $|\psi_k\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} \omega_r^{jk} |a^j\rangle$ est vecteur propre de U_a pour une valeur propre à calculer

Factoriser

- Montrer qu'en partant d'une superposition uniforme de vecteurs propres $|\psi_k\rangle$ de U_a, alors l'estimation de valeur propre avec m bits de précision fourni y tq $y/2^m \approx k/r$
- Quelle est la superposition uniforme des vecteurs propres $|\psi_k\rangle$ de U_a
- Conclure avec un autre algorithme pour la factorisation. Donner sa complexité.

Définition

- $G = (V,E)$ un graphe (non orienté) sur les sommets V d'arêtes E
- Une marche aléatoire est un déplacement aléatoire
  - sur les sommets V de G
  - en suivant les arêtes E de G
  - tel que $\Pr[u \to v] = 1/\deg(u)$



Application typique

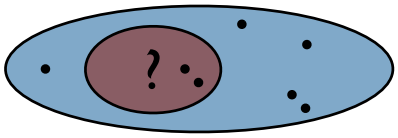
- Recherche de chemin de s à t

Mélange

- Comment mélanger un jeux de cartes et en combien de temps ?

Estimation

- Comptage, surface, volume
- Page rank de Google



Parcours

- Comment envoyer un message au Dalai Lama, le plus rapidement possible ? (pas de recours à l'annuaire)
- Quel est le comportement d'une particule aléatoire
  - sur une ligne, dans le plan, l'espace, ... ?
- Comment sortir d'un labyrinthe avec peu de mémoire ?

Modéliser/Analyser des comportements

- Algorithmes probabilistes
- Réseaux, Jeux, Finance

Recherche

- Comment trouver une information dans une base de données gigantesque

Définition

- Un processus stochastique (discret à espace fini) $\boldsymbol{X}$ est une suite de variables aléatoires X_t indexée par un paramètre *temps* (discret, $t = 0, 1, 2, 3, ...$) (espace fini, $\{X_t\}_t$ prennent un nombre fini de valeurs)
- Une chaîne de Markov est un processus stochastique tel que

  $\Pr[X_t = a_t \mid X_{t-1} = a_{t-1}, X_{t-2} = a_{t-2}, \dots, X_0 = a_0]$

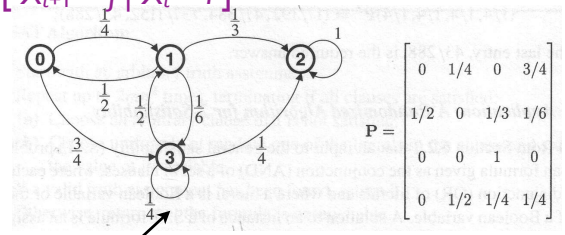
  $= \Pr[X_t = a_t \mid X_{t-1} = a_{t-1}]$ pas d'effet mémoire

  $= P(a_t, a_{t-1})$ indépendant du temps

  en résumé l'évolution ne dépend que de la valeur courante
- Matrice de transition : $P(i,j) = \Pr[X_{t+1} = j \mid X_t = i]$

Remarque.

- Etat de X_t : $P_t(i) = \Pr[X_t = i]$
- $P_t = P_0 \times P^t$



Exemples

- Marches aléatoires
- Marches aléatoires avec des boucles : $P' = 1/2\ P + 1/2\ \mathrm{Id}$
- Somme convexe de marches aléatoires : $P'' = 3/4\ P + 1/4\ P'$

SAT

- Entrée : suite de clauses sur n variables

  x_1 ou $\overline{x_2}$ ou $\overline{x_5}$

  $\overline{x_4}$ ou $\overline{x_3}$ ou x_2
- Sortie : une assignation $a \in \{0,1\}^n$ qui satisfait **toutes** les clauses

k-SAT

- Restriction : chaque clause utilise au plus k variables

Théorème

- 3-SAT est NP-complet
- 2-SAT est résoluble en temps linéaire et espace linéaire
- 2-SAT est résoluble en temps quadratique et espace logarithmique

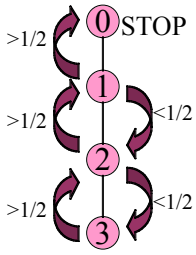
Algorithme

- Choisir une assignation quelconque a
- Répéter $2m \times n^2$ fois, et s'arrêter si a satisfait toutes les clauses
  - Choisir une clause C arbitraire non satisfaite
  - Choisir aléatoirement une des variables x_i de C
  - Changer la valeur du bit i de a correspondant à cette variable
- Si a satisfait toutes les clauses, renvoyer a
- Sinon renvoyer que toutes les clauses ne sont satisfiables simultanément

Analyse de l'algorithme

- Fixer une solution (potentielle) s
- X_t = nombre de bits différents entre a et s à la t-ème itération
- Nécessairement

$$\begin{aligned}\Pr[X_{t+1} = n-1 | X_t = n] &= 1\\ \Pr[X_{t+1} = 0 | X_t = 0] &= 1\\ \Pr[X_{t+1} = j-1 | X_t = j] &\geq 1/2\end{aligned}$$



Suite de l'analyse

- Version pessimiste du comportement de l'algorithme

$$\begin{aligned}Y_0 &= n\\ \Pr[Y_{t+1} = n-1 | Y_t = n] &= 1\\ \Pr[Y_{t+1} = 0 | Y_t = 0] &= 1\\ \Pr[Y_{t+1} = j-1 | Y_t = j] &= 1/2\end{aligned}$$

- h_j : temps moyen d'atteindre l'état 0 en partant de l'état j

$$\begin{aligned}h_n &= 1 + h_{n-1}\\ h_0 &= 0\\ h_j &= \frac{h_{j-1} + h_{j+1}}{2} + 1\end{aligned}$$

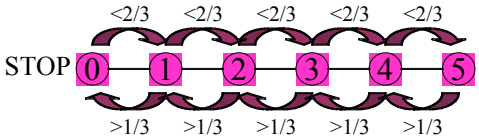
- La résolution donne pour $j \neq 0$

$$h_j = h_{j-1} + 2(n-j) + 1$$

- Et donc $h_n = \sum_{i=0}^{n} (2i+1) = n^2$
- D'après l'inégalité de Markov, La probabilité de ne pas trouver de solution après $2n^2$ étapes est au plus $1/2$, et après $2mn^2$ étapes est au plus $1/2^m$

Même algorithme :

- Analyse montre qu'on a tendance à s'éloigner d'une solution



Alors ?

- Idée 1 : partir d'une assignation aléatoire
- Idée 2 : recommencer à partir d'une assignation aléatoire si aucune solution n'est trouvée au bout de $3n$ itérations

Théorème

- L'algorithme ainsi modifié trouve une solution en temps moyen $(4/3)^n$

Remarques

- Mieux que toutes les autres approches déterministe, dans le pire cas
- Toutes les marches aléatoires pour 3-SAT suivent la même structure

Définition

- Une distribution stationnaire est une distribution de probabilité π tq $\pi = \pi P$

Définition

- Une marche aléatoire est ergodique si elle est définie sur un graphe
  - connexe : aucun sommet déconnecté du graphe
  - non biparti : les arêtes ne sont pas uniquement entre deux sous-parties disjointes

Remarque : L'ergodicité se généralise aux chaînes de Markov...

Théorème

- Toute chaîne de Markov ergodique possède une unique distribution stationnaire π

Temps de mélange

- Définition : $\text{Max}_x \{$ minimum T tq $P^T(x,i) \approx \pi_i\}$
- Spectral gap : δ = 1 – 2e plus grande |valeur propre|
- Prop: P marche aléatoire → temps de mélange en $O(1/\delta \times \log(1/\min_x \pi_x))$
  (se généralise au chaînes de Markov réversible)

Some examples

- [Grover'95]: Complete graph
- [Shenvi, Kempe, Whaley'03]: Hypercube
- [Ambainis'04]: Johnson Graph
- [Ambainis, Kempe, Rivosh'05] [Tulsi'08]: 2D-Grid

Generic construction

- Quantum analogue $W(P)$ of a symmetric Markov chain P [Szegedy'04]
- Theorem [Szegedy'04]

  Phase gap $\Delta(W(P)) = \sqrt{\delta(P)}$
- Remain valid for reversible walks in [MNayakRolandSantha'07]

Continuous time

- Hamiltonian = Adjacency matrix of the graph
- Discretization using **Phase estimation** [Childs'08]

Input

- Set $X = \{a, b, c, \ldots\}$
- Marked elements M subset of X

Output

- Some marked element x in M

Cost

- **Checking**: answering "x in M ?"

Exhaustive search

- Cost = $|X|$ x **Checking**

Randomized search

- **Setup**: sampling according to a distribution π
- Cost = $1/\varepsilon$ x (**Checking** + **Setup**) $\quad \varepsilon = \Pr_\pi(M)$

Quantum search [Grover'96, ...]

- Cost = $1/\sqrt{\varepsilon}$ x (**Checking** + **Setup**)

Starting state

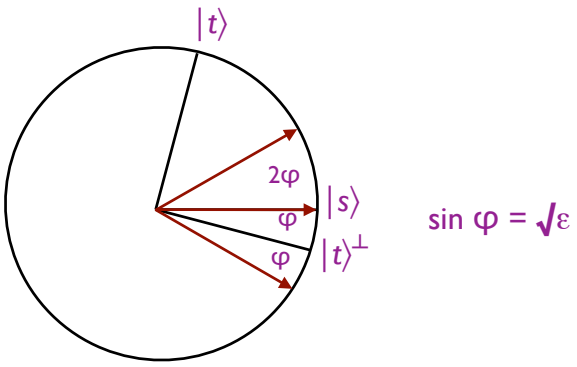
- $|s\rangle$: built using **Setup**: $|0\rangle \rightarrow |s\rangle$

Target state

- $|t\rangle$: projection of $|s\rangle$ to marked elements

Alternately reflect through $|t\rangle^{\perp}$ and $|s\rangle$

- $|\langle s|t\rangle|^2 = \varepsilon$



- # of iterations: $\Theta(1/\sqrt{\varepsilon})$

Reflection through $|t\rangle^{\perp}$

- Using **Checking**: Flip amplitudes of marked elements

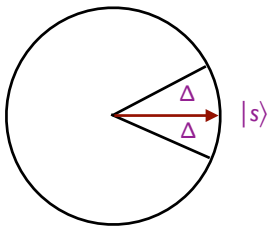
Reflection through $|s\rangle$

- A la Grover
  - Undo **Setup** $|s\rangle \rightarrow |0\rangle$
  - Reflect through initial state: $|0\rangle \rightarrow |0\rangle \quad |\psi\rangle \rightarrow -|\psi\rangle$
  - Do **Setup** $|0\rangle \rightarrow |s\rangle$

Complexity

- # of iterations: $1/\sqrt{\varepsilon}$
- Each step: **Checking** + **Setup**
- Total: **Setup** + $1/\sqrt{\varepsilon}$ x (**Checking** + **Setup**)

Approximate reflexion through $|s\rangle$ by unitary W



- Cost: **Update**
- Unique 1-eigenvector (i.e. phase 0): $|s\rangle$
- Phase gap: Δ

Previous approach [Ambainis'04]

- If Δ is constant, then W *simulates* Reflection through $|s\rangle$
- Hope: $W^{1/\Delta}$ have constant phase gap

Phase estimation approach [MNayakRolandSantha'07]

1. Estimate W-eigenphase of current state with precision Δ
2. If phase $\neq 0$, flip amplitude
3. Undo Step 1

Complexity

- Each step: **Checking** + 2(# of calls to W in Step 1) x **Update**

Goals

- Efficient implementation of Phase estimation
- Construct W such that: (# of calls to W in Step 1) x **Update** << **Setup**

δ = spectral gap
ε = success probability
P = reversible Markov chain

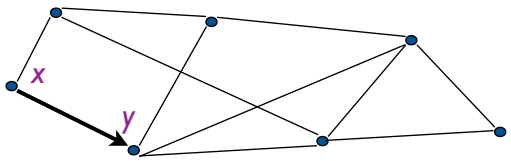
Random search

1. Start from stationary distribution
2. Repeat for $1/\varepsilon$ steps
   a. Check if current state is marked
   b. Simulate $1/\delta$ steps of P
3. If no marked element is found, output "none marked"

Theorem

- **Random search** outputs a marked element with cost

  Setup + $1/\varepsilon \times (\ 1/\delta \times$ **Update** + **Checking** $)$



G : connected graph on $\{1, 2, ..., n\}$

$P = \begin{pmatrix} & \vdots \\ x & p_{xy} \\ & \end{pmatrix}$ (y)

Walk on edges

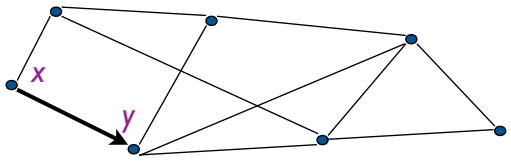
- States: edges (x,y) "from x to y"
- One step

  Sample y in neighborhood of x ("Coin Flip"): apply $F^x = (P_x,...,P_x)$ to y

  Exchange roles between origin and destination ("Move"): Swap

- Global operator: $\mathbf{Swap} \times (\sum_x |x\rangle\langle x| \otimes F^x)$

  defined over distributions on edges of G



G : connected graph on $\{1, 2, ..., n\}$

$P = \begin{pmatrix} & \vdots \\ x & p_{xy} \\ & \end{pmatrix}$ (y)

Quantum walk on G

- Amplitude transitions given by a family of unitaries $(F^x)_x$

  defined on H^x = Span ($|y\rangle$: (x,y) edge of G)
- One step from $|x\rangle|y\rangle$ in G

  Apply F^x to second register $|y\rangle$

  Swap two registers
- Global operator: $W = \mathbf{Swap} \times (\sum_x |x\rangle\langle x| \otimes F^x)$

  defined on $H(G)$ = Span ($|x,y\rangle$: (x,y) edge of G)

From random walk P to quantum walk $W(P)$

- Set $F_x = 2|p_x\rangle\langle p_x| - 1$, where $|p_x\rangle = \sum_y \sqrt{p_{xy}}|y\rangle$
- Stationary state $|s\rangle = \sum_x \sqrt{\pi_x}|x\rangle|p_x\rangle$, if π stationary distribution of P
- Phase gap $\Delta(W(P)) = \sqrt{\delta(P)}$

δ = spectral gap

ε = success probability

P = reversible Markov chain

Find [MNayakRolandSantha'07]

1. Start from stationary superposition
2. Repeat for $1/\sqrt{\varepsilon}$ steps
   a. Flip amplitude if current state is marked
   b. Perform a reflection through stationary superposition using
      Phase estimation for $W(P)$ to current state with precision $\sqrt{\delta}$
3. Measure: If not marked, output "none marked", else output state

Theorem

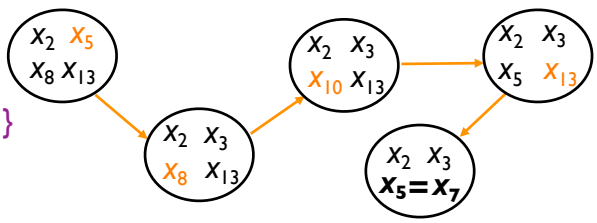
- **Find** outputs a marked element with cost
  Setup + $1/\sqrt{\varepsilon}$ × ($1/\sqrt{\delta}$ × **Update** + **Checking**)

Element Distinctness

- Input: List of n numbers $\{x_1, x_2, x_3, \ldots, x_n\}$
- Output: Are all the numbers distinct?
  (or is there a collision: $x_i = x_j$, $i \neq j$)
- Complexity: # of queried numbers
- Classically: n
- Using Grover search: $n^{3/4}$ [BuhrmanDurrHeiligmanHoyerMSanthaWolf'01]
- Using quantum walk on Johnson graph: $n^{2/3}$ [Ambainis'04]
- Lower bound: $n^{2/3}$ [Aaronson'02, Shi'02]

Algorithm

- Johnson Graph (n, r)
  Vertices: $\{ S \subseteq \{1, 2, \ldots, n\} : \#S=r \}$
  Edges: $\{ (S, T) : \#T \cap S = n-1 \}$

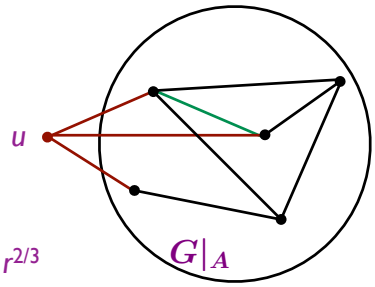


- Spectral gap: $\delta = O(1/r)$
- Success probability: $\varepsilon = \Pr[$ collision in random $S] \approx (r/n)^2$
- Runtime: $r + (n/r)^{2/2} (\sqrt{r} \times 2 + 0) \rightarrow n^{2/3}$
  setup, update, checking

Triangle Finding

- Input: Graph on n nodes
- Output: Is there a triangle in the graph?
- Complexity: # of queried edges
- Classically: n^2
- Using Grover search: $n^{3/2}$
- Using two recursive walks on Jonson graph: $n^{1.3}$ [MSanthaSzegedy'05]
- Lower bound: n

Algorithm



- Fix subset A of r vertices and vertex u
- Query $G|_A$: r^2
- Decide if $G|_A$ makes a triangle with vertex u: $r^{2/3}$
- Find u s.t. " " " : $\sqrt{n} \times r^{2/3}$
- Decide if $G|_A$ contains a triangle edge : $\sqrt{n} \times r^{2/3}$
- Find a triangle edge : $r^2 + (n/r)(\sqrt{r} \times r + \sqrt{n} \times r^{2/3})$
- Find the 3rd vertex of a triangle in G : $\sqrt{n}$

Conclusion: $r = n^{3/5} \rightarrow n^{1.3}$