# Algorithmes quantiques variationnels améliorés par Chaînes de Markov Monte-Carlo

Mohammed Adam Khali , Aymane Dhimen & Marouane R'bib

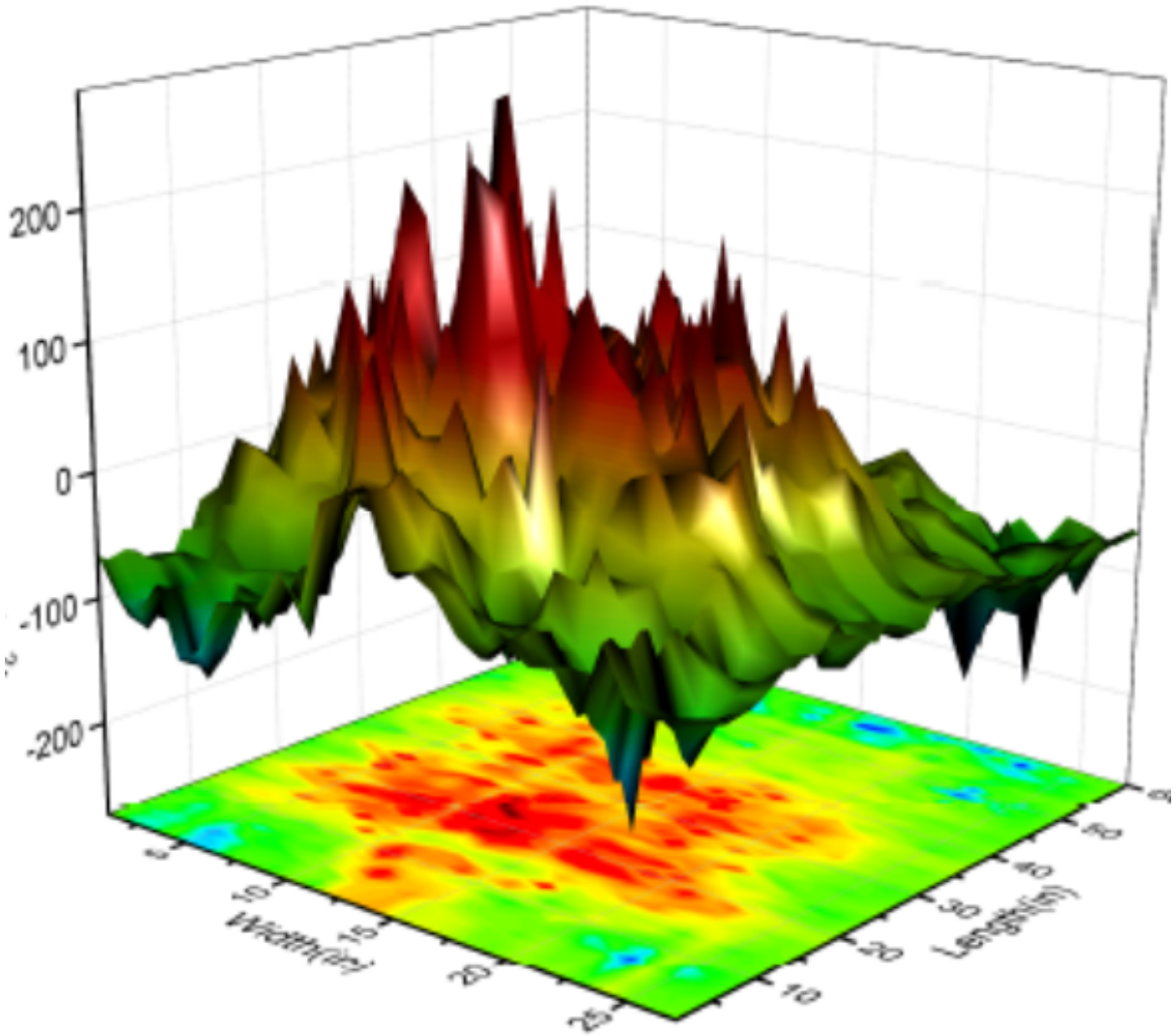
Decembre 2023

# Contents


0.1 Introduction . . . . . 2
0.2 Le problème . . . . . 5
0.3 Algorithme quantique variationnel - Chaînes de Markov Monte Carlo . . 5
0.3.1 Algorithme MCMC-VQA . . . . . 6
0.3.2 Obtention de la distribution Boltzmann . . . . . 7
0.3.3 Calcul de la distribution de proposition . . . . . 7
0.3.4 Distribution d'acceptation et proposition . . . . . 8
0.4 Implémentation sur un processeur quantique . . . . . 9
0.5 Preuve d'ergodicité . . . . . 12
0.6 Conclusion . . . . . 12

# I Introduction

La résolution de problèmes d'optimisation complexe constitue un défi majeur dans divers domaines, allant de la combinatoire classique à la chimie quantique et à la physique de la matière condensée. Parmi les approches innovantes émergentes, les algorithmes quantiques variationnels (VQE) offrent des perspectives prometteuses pour aborder ces défis. Cependant, la convergence vers les minima globaux et la performance de VQE peuvent être entravées par la présence de minima locaux. Dans ce contexte, nous présentons algo-




rithme, MCMC-VQA, qui combine habilement les techniques de chaînes de Markov Monte Carlo (MCMC) avec les algorithmes quantiques variationnels pour résoudre le problème classique de MaxCut. Le problème de MaxCut, consistant à optimiser la découpe d'un graphe non orienté, est formulé mathématiquement pour maximiser la somme pondérée des arêtes coupées. Nous utilisons l'algorithme VQE pour encoder le graphe dans le Hamiltonien du modèle d'Ising, et introduisons la méthode MCMC pour surmonter les limitations liées aux minima locaux.

Cette approche hybride, MCMC-VQA, explore les propriétés ergodiques pour garantir la convergence vers les solutions globales. Nous évaluons l'efficacité de notre algorithme à travers des simulations numériques, démontrant sa capacité à surpasser les méthodes

traditionnelles en surmontant les obstacles des minima locaux.

Cette étude offre une perspective novatrice sur la résolution de problèmes d'optimisation difficile en combinant judicieusement des techniques quantiques et classiques. Nous présentons les détails de l'implémentation sur un processeur quantique et fournissons une preuve d'ergodicité pour renforcer la robustesse de notre approche.

# Définitions

MaxCut est un problème de partitionnement sur des graphes non orientés $G$ (Fig. 1, noir), où les arêtes $\omega_i$ relient des paires de sommets $v_{ia}, v_{ib}$ . L'objectif est d'assigner de manière optimale toutes les valeurs de sommets $v_{ia}, v_{ib} \in \{-1, 1\}$, afin de maximiser la fonction objectif.

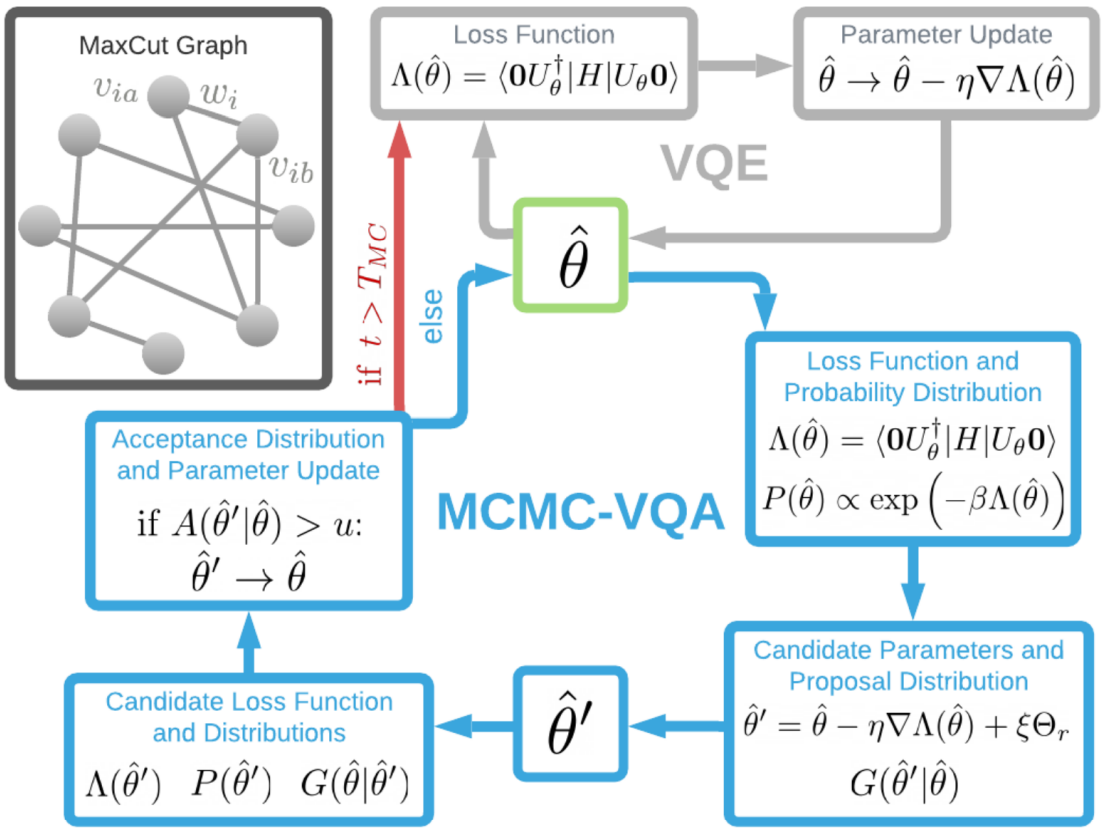



Figure 1: Diagramme d'un graphe aléatoire pour MaxCut, VQE et MCMC-VQA

Les algorithmes quantiques variationnels (VQE) sont des méthodes d'optimisation destinées à résoudre des problèmes complexes dans des domaines tels que la combinatoire classique, la chimie quantique et la physique de la matière condensée. Cette approche repose sur la programmation quantique, mais l'optimisation de ces algorithmes peut être complexe en raison de paysages non convexes. Dans cette étude, nous introduisons une amélioration du VQE qui intègre des techniques classiques de chaînes de Markov Monte Carlo (MCMC) pour converger de manière prouvée vers les solutions globales des problèmes d'optimisation, offrant ainsi des performances améliorées et garanties.

# II Le problème

Dans ce projet, nous nous intéressons à la résolution du problème de MaxCut via l'algorithme quantique variationnel (VQE) et son amélioration par l'utilisation de la méthode de chaînes de Markov Monte Carlo (MCMC). Le problème de MaxCut consiste à optimiser la découpe d'un graphe de manière à maximiser la somme des poids des arêtes coupées. Formellement, le problème est formulé comme suit :

$$\text{maximiser} \quad \frac{1}{2}\sum_i w_i(1 - v_i v_i^b), \tag{1}$$

où :

- $w_i$ : le poids de l'arête $i$,
- $v_i$ : la valeur du sommet $i$,
- $v_i^b$ : la valeur du sommet associé à $i$ dans la paire $(v_i, v_i^b)$ que l'on souhaite optimiser.

Cela exprime de manière mathématique l'objectif de maximiser la somme pondérée des termes $1 - v_i \cdot v_i^b$, où $v_i$ et $v_i^b$ sont pris dans l'ensemble $\{-1, 1\}$.

Pour résoudre MaxCut via VQE, nous encodons le graphe $G$ dans le Hamiltonien du modèle d'Ising, défini comme :

$$H = \sum_i \omega_i \sigma_{ia} \sigma_{ib}, \tag{2}$$

où $\omega_i$ provient de la fonction objectif de MaxCut, et via, vib $\rightarrow \sigma_{ia}, \sigma_{ib}$ pour les opérateurs de spin Pauli-Z $\sigma_{ia}$, $\sigma_{ib}$. Maximiser la découpe de $G$ est alors équivalente à minimiser la fonction de perte $\Lambda_t$, définie comme :

$$\Lambda_t = \Lambda(\hat{\theta}_t) = \langle 0|(U_t^\dagger H U_t)|0\rangle = \sum_i \omega_i \langle \sigma_{ia}\sigma_{ib}\rangle_t, \tag{3}$$

où $\hat{\theta}_t$ représente les paramètres variables appris pendant l'entraînement du circuit quantique. Cependant, la convergence vers les minima globaux et la performance de VQE peuvent être limitées.

Dans ce contexte, nous introduisons la méthode MCMC-VQA, qui combine la descente de gradient classique avec des techniques de chaînes de Markov Monte Carlo pour améliorer et garantir la performance des algorithmes quantiques variationnels.
Nous explorons les propriétés ergodiques de cette méthode et évaluons son efficacité à travers des simulations numériques. Cette méthode est particulièrement importante, car le gradient de toute variable $\theta_k^t \in \hat{\theta}_t$ peut être calculé comme $\nabla_k \Lambda(\hat{\theta}_t) = \frac{\Lambda(\hat{\theta}_t + \epsilon\hat{k}) - \Lambda(\hat{\theta}_t - \epsilon\hat{k})}{2\epsilon}$ par différence finie. Comme $\nabla\Lambda(\hat{\theta}_t) \rightarrow 0$ à proximité des minima globaux et locaux, l'entraînement VQE est sujet à la stagnation à des solutions sous-optimales.

# III Algorithme quantique variationnel - Chaînes de Markov Monte Carlo

Pour obtenir $\pi$ pour une distribution d'intérêt, Metropolis-Hastings spécifie le noyau de transition $P(x'|x)$, qui est la probabilité que l'état $x$ passe à l'état $x'$. Généralement,

le processus markovien est défini de telle sorte que les transitions satisfont la condition d'équilibre détaillé :

$$\mu(x)P(x'|x) = \mu(x')P(x|x') \tag{4}$$

Lorsque l'équation 4 est satisfaite, on dit que la chaîne est réversible et est garantie de converger vers une distribution stationnaire. $P(x'|x)$ peut être décomposé en deux quantités :

$$P(x'|x) = G(x'|x)A(x'|x),$$

où $G(x'|x)$ est la distribution de proposition, ou la probabilité conditionnelle de proposer l'état $x'$ étant donné l'état $x$, et $A(x'|x)$ est la distribution d'acceptation, ou la probabilité d'accepter le nouvel état $x'$ étant donné l'état $x$. Pour satisfaire l'équation 4, la distribution d'acceptation est définie comme :

$$A(x'|x) = \min\left(1, \frac{\mu(x')G(x|x')}{\mu(x)G(x'|x)}\right).$$

Notez que, comme seule la relation $\frac{\mu(x')}{\mu(x)}$ est considérée, la distribution de probabilité n'a pas besoin d'être normalisée. Pour déterminer si l'état candidat $x'$ ou l'état actuel $x_t$ devrait être utilisé comme état futur $x_{t+1}$, un échantillon $u$ est tiré de la distribution uniforme $U(0,1)$, et la décision est prise selon la règle suivante :

$$x_{t+1} = \begin{cases} x' & \text{si } u \leq A(x' \mid x_t), \\ x_t & \text{sinon.} \end{cases} \tag{5}$$

La procédure de Metropolis-Hastings garantit que la distribution stationnaire de cette chaîne de Markov est la distribution de probabilité souhaitée $\pi$. En d'autres termes, après un nombre suffisamment grand d'itérations, les états générés par la chaîne de Markov suivront la distribution $\pi$.

### 0.3.1 Algorithme MCMC-VQA

L'algorithme MCMC-VQA combine les techniques de chaînes de Markov Monte Carlo (MCMC) avec les algorithmes d'optimisation variationnelle quantique (VQA) pour résoudre le problème MaxCut. Voici un aperçu de l'algorithme :

1. Initialiser les paramètres quantiques $\hat{\theta}_0$ de manière aléatoire.
2. Répéter les étapes suivantes pour un certain nombre d'épisodes MCMC (processus markovien) :
   (a) Proposer un nouvel état quantique $\hat{\theta}'$ selon la distribution de proposition $G(\hat{\theta}'|\hat{\theta}_t)$.
   (b) Calculer la distribution d'acceptation $A(\hat{\theta}'|\hat{\theta}_t)$.
   (c) Générer un échantillon $u \sim U(0,1)$.
   (d) Mettre à jour l'état quantique selon la règle de Metropolis-Hastings.
3. Initialiser une série d'épisodes traditionnels de VQA avec les meilleurs paramètres $\hat{\theta}_{\min}$ trouvés pendant le processus MCMC.

4. Effectuer une séquence de clôture des épisodes de VQA pour converger rapidement vers le minimum local.

5. Retourner les paramètres quantiques $\hat{\theta}_{\min}$ obtenus.

La figure 2 illustre des exemples de trajectoires générées par MCMC-VQA avec différentes températures thermodynamiques inverses. Les trajectoires suivent le processus markovien pendant un certain nombre d'épisodes, puis convergent rapidement vers le minimum local pendant les épisodes de clôture VQA.

L'implémentation sur un processeur quantique réalise des mesures pour estimer la fonction de perte $\Lambda(\hat{\theta})$. Les variances nécessaires pour la distribution d'acceptation sont estimées à partir des déviations standard des statistiques de la valeur attendue. L'algorithme MCMC-VQA garantit l'ergodicité de la chaîne de Markov, explorant efficacement l'espace des solutions pour converger vers le minimum global.

### 0.3.2 Obtention de la distribution Boltzmann

Dans le cadre de la résolution du problème MaxCut via VQE, nous définissons $P(\hat{\theta})$ comme la distribution de Boltzmann [**?**] :

$$P(\hat{\theta}_a) = \frac{e^{-\beta\Lambda_a}}{Z}, \quad Z = \sum_i e^{-\beta\Lambda_i},$$

où $\Lambda_a$ est la valeur propre associée à l'état, et $\beta$ est l'inverse de la température thermodynamique. La probabilité d'un état augmente exponentiellement avec la diminution de la fonction de perte.

### 0.3.3 Calcul de la distribution de proposition

Pour calculer la distribution de proposition $G(\hat{\theta}'|\hat{\theta}_t)$, nous prenons en compte les statistiques d'échantillonnage des VQA en raison de l'incertitude quantique. En raison de cette incertitude, la mesure $m_{ri}(\hat{\theta}_t)$ des opérateurs $\omega_i\sigma_{ia}\sigma_{ib}$ est un échantillon d'une distribution avec une moyenne $\mu_{it}$ et une variance $(\Omega_{it})^2$ :

$$(\Omega_{it})^2 = \omega_i^2 \left[\langle(\sigma_{ia}\sigma_{ib})^2\rangle_t - \langle\sigma_{ia}\sigma_{ib}\rangle_t^2\right] = \omega_i^2 \left[1 - (\mu_{it})^2\right].$$

Selon le Théorème Central Limite, en supposant au moins $M \geq 30$ mesures indépendantes et identiquement distribuées $m_{ri}(\hat{\theta}_t)$, une estimation de la fonction de perte $\Lambda_t$ est la statistique $l_t \sim \mathcal{N}(\Lambda_t, (\Omega_{\Lambda t})^2)$, où $(\Omega_{\Lambda t})^2 = \sum_i (\Omega_{it})^2/M$.

De manière similaire, pour tout $\theta_k^t \in \hat{\theta}_t$ et en supposant de petites variations de paramètres $\varepsilon$, le gradient $\nabla_k\Lambda_t = \frac{1}{2\varepsilon}\left[\Lambda(\hat{\theta}_t + \varepsilon\hat{k}) - \Lambda(\hat{\theta}_t - \varepsilon\hat{k})\right]$ est la statistique $d_k l_t \sim \mathcal{N}(\nabla_k\Lambda_t, [\Omega_\Lambda^2(\hat{\theta}_t + \varepsilon\hat{k}) + \Omega_\Lambda^2(\hat{\theta}_t - \varepsilon\hat{k})]/(4\varepsilon^2)$. La variance de cette distribution peut être simplifiée en notant que, jusqu'à l'ordre $\varepsilon$, les opérateurs de Pauli décalés de paramètre sont $\sigma_{ia}^{\pm k} = \sigma_{ia}(\hat{\theta} \pm \varepsilon\hat{k}) = \sigma_{ia} \pm \iota_{iak}$, où $\sigma_{ia} = \sigma_{ia}(\hat{\theta})$ et $\iota_{iak} = \left(\frac{\partial\sigma_{ia}}{\partial\theta^k}\right)\varepsilon$. Nous pouvons ensuite simplifier la somme $\Omega_i(\hat{\theta}_t + \varepsilon\hat{k})^2 + \Omega_i(\hat{\theta}_t - \varepsilon\hat{k})^2 = 2\Omega_i(\hat{\theta}_t)^2$. À présent, jusqu'à l'ordre $\iota$, nous pouvons dériver le gradient de la distribution

$$d_k l_t \sim \mathcal{N}\left(\nabla_k\Lambda_t, \frac{\Delta^2\Lambda(\hat{\theta}_t)}{2\varepsilon^2}\right). \tag{6}$$

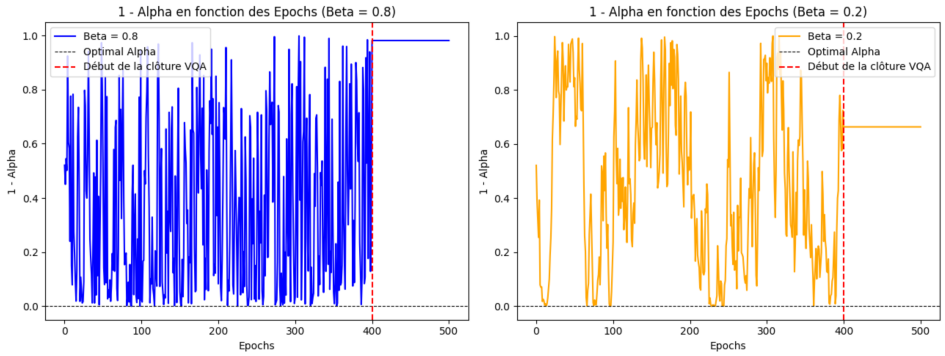



Figure 2: Exemples de trajectoires avec une température thermodynamique inverse $\beta = 0.8$ (gauche) et $\beta = 0.2$ (droite). Quatre cents épisodes MCMC-VQA (épisodes markoviens) sont suivis d'une séquence de clôture des épisodes de VQE (commençant à la ligne rouge en pointillés), qui est initialisée avec les meilleurs paramètres $\hat{\theta}_{\min}$ trouvés pendant le processus markovien. À basse température ($\beta = 0.8$), les trajectoires restent piégées dans les minima locaux, et atteindre l'ergodicité est un processus long. En revanche, les trajectoires à haute température ($\beta = 0.2$) atteignent rapidement la phase d'initialisation, générant $\hat{\theta}_{\min}$ qui conduisent à une convergence presque parfaite pendant la séquence de clôture VQE. Voir la section III pour les détails de la simulation.

La descente de gradient standard proposerait l'état candidat $\hat{\theta}' = \hat{\theta} - \eta \nabla \Lambda_t$, cependant MCMC-VQA ajoute un terme de bruit aléatoire normalement distribué $\Theta_r \sim \mathcal{N}(0, 1)$ avec un paramètre d'échelle $\xi$ afin d'élargir le support de la distribution de proposition $G(\hat{\theta}'|\hat{\theta}_t)$. Cela spécifie que

$$G(\hat{\theta}'|\hat{\theta}_t) = \prod_k G(\hat{\theta}'|\hat{\theta}_t)_k, \quad G(\hat{\theta}'|\hat{\theta}_t)_k = \mathcal{N}\left(\eta \nabla_k \Lambda(\hat{\theta}_t), \xi^2 + \eta^2 \left(\Delta \Lambda_t\right)^2 \frac{1}{2\varepsilon^2}\right)(\hat{\theta}_t - \hat{\theta}')$$

Il en découle que la distribution d'acceptation est donnée par

$$A(\hat{\theta}'|\hat{\theta}_t) = \min\left(1, \frac{P(\hat{\theta}')G(\hat{\theta}_t|\hat{\theta}')}{P(\hat{\theta}_t)G(\hat{\theta}'|\hat{\theta}_t)}\right)$$

Nous notons que $G(\hat{\theta}_t|\hat{\theta}')$ est obtenue en échangeant simplement $\hat{\theta}_t$ et $\hat{\theta}'$ dans l'équation 11. Un échantillon uniforme aléatoire $u \sim U(0, 1)$ est ensuite tiré pour la comparaison, de sorte que $\hat{\theta}_{t+1} = \hat{\theta}'$.

### 0.3.4 Distribution d'acceptation et proposition

La distribution d'acceptation $A(\hat{\theta}'|\hat{\theta}_t)$ est déterminée par la comparaison entre l'état candidat $\hat{\theta}'$ et l'état actuel $\hat{\theta}_t$. Un échantillon uniforme aléatoire $u \sim U(0, 1)$ est alors tiré, et si $A(\hat{\theta}'|\hat{\theta}_t) \geq u$, alors $\hat{\theta}_{t+1} = \hat{\theta}'$ et l'état candidat est accepté. Sinon, $\hat{\theta}_{t+1} = \hat{\theta}_t$ et l'état candidat est rejeté.

Après $TMC$ épisodes du processus markovien ci-dessus, MCMC-VQA implémente une courte série d'épisodes traditionnels de VQA pour une convergence rapide vers le minimum le plus proche. Ces épisodes de VQA de clôture sont notamment initialisés avec $\hat{\theta}_{\min}$, l'ensemble de paramètres de la plus basse valeur propre $\Lambda_{\min}$ trouvée pendant la phase Metropolis-Hastings. De cette manière, MCMC-VQA peut être considérée

comme une procédure de "démarrage chaud" , mais avec des garanties ergodiques.
Les trajectoires exemplaires de MCMC-VQA sont présentées à la Fig. 2 avec des températures thermodynamiques inverses $\beta = 0.8$ et $\beta = 0.2$. Les détails de toutes les simulations sont fournis dans la section III. Notre algorithme combine l'optimisation basée sur la descente de gradient de VQE avec un processus markovien qui échappe aux minima locaux. Une telle exploration est significativement plus importante à la température plus élevée $\beta = 0.2$, où, plutôt que de se stabiliser dans des bassins de fonction de perte distincts d'où l'évasion est relativement rare, les trajectoires présentent le comportement caractéristique de "burn-in" des chaînes de Markov ergodiques. Au moment où les époques de clôture de VQE sont appliquées, les chaînes MCMC-VQA ergodiques $\beta = 0.2$ ont échantillonné des états suffisamment proches du minimum global et convergent vers la vérité au sol presque uniformément.

La Figure 2 (à gauche) affiche la précision moyenne $1-\alpha$ et l'écart type des solutions MaxCut avec MCMC-VQA en fonction de $\beta$. Les lignes en pointillés représentent les performances de la VQE traditionnelle sur le même ensemble de graphes et d'ansatz de circuit. Nous notons que toutes les valeurs simulées de $\beta$ surpassent la VQE traditionnelle. Jusqu'à $\beta \sim 0.2$, les chaînes MCMC-VQA à température plus élevée ont une précision plus élevée et une meilleure convergence, car leur paramètre de température plus permissif oriente la distribution d'acceptation vers l'acceptation des états candidats. Cependant, les performances diminuent à des températures très élevées, pour lesquelles les chaînes MCMC-VQA ne sont plus sensiblement orientées vers la minimisation de l'énergie, et l'algorithme devient plus similaire à un échantillonnage aléatoire qu'à une descente de gradient intrépide. De même, la quantité optimale de bruit de mise à jour des paramètres $\xi$ est inversement proportionnelle à $\beta$ (2), car des températures plus élevées permettent des écarts plus radicaux par rapport à la descente de gradient standard.

# IV Implémentation sur un processeur quantique

Comme mentionné précédemment, la fonction de perte $\Lambda_t$ n'est pas précisément déterminée sur un processeur quantique réel, mais est plutôt estimée sous la forme d'une statistique $l_t = \frac{1}{M}\sum_{r=1}^{M} m_{ri}(\hat{\theta}_t)$, où $m_{ri}(\hat{\theta}_t)$ est une mesure quantique. Par conséquent, la variance d'une seule mesure d'observable $(\Omega_{it})^2$ est estimée par $(\Omega_{it})^2 = \omega_i^2[1-(q_{it})^2]$, tandis que la variance de la fonction de perte totale $(\Omega_{\Lambda t})^2$ est estimée par $(\Omega_{\Lambda t})^2 = \sum_i (\Omega_{it})^2/M$.

Alternativement, les variances pourraient être estimées directement à partir des déviations standard des statistiques de la valeur attendue. Nous définissons alors la distribution d'acceptation sur le matériel quantique comme suit :

$$A(\hat{\theta}'|\hat{\theta}_t) = \min\left(1, \frac{p(\hat{\theta}')g(\hat{\theta}_t|\hat{\theta}')}{p(\hat{\theta}_t)g(\hat{\theta}'|\hat{\theta}_t)}\right),$$

où $p(\hat{\theta}) \propto e^{-\beta l_t}$, $g(\hat{\theta}_t|\hat{\theta}') = \prod_k g(\hat{\theta}_t|\hat{\theta}')_k$, et

$$g(\hat{\theta}_t|\hat{\theta}')_k = N\left(\eta d_k l_t, \xi^2 + \eta^2(\Omega_{\Lambda t})^2\frac{1}{2\varepsilon^2}\right)(\hat{\theta}_t - \hat{\theta}'),$$

Il s'ensuit que la distribution d'acceptation est donnée par

$$A(\hat{\theta}'|\hat{\theta}_t) = \min\left(1, \frac{e^{-\beta l_{t'}} \prod_k e^{-\frac{1}{2}\left(\frac{\hat{\theta}'_k - \hat{\theta}_k}{\eta dklt}\right)^2}}{e^{-\beta l_t} \prod_k e^{-\frac{1}{2}\left(\frac{\hat{\theta}_k - \hat{\theta}'_k}{\eta dklt}\right)^2}}\right),$$

où $l_{t'} = \frac{1}{M}\sum_{r=1}^{M} m_{ri}(\hat{\theta}')$. MCMC-VQA n'augmente pas la complexité quantique des VQA (nombre d'opérations effectuées sur le matériel quantique), car les mesures pour estimer $\Lambda(\hat{\theta})$ sont effectuées de la manière habituelle. De plus, la distribution d'acceptation et ses composants sont calculés classiquement avec une arithmétique simple. Voici un exemple de code Python illustrant l'algorithme MCMC-VQA mentionné ci-dessus :

```python
import numpy as np
import matplotlib.pyplot as plt

# Fonction objectif de MaxCut
def maxcut_objective(vertices, vertices_b, weights):
    return 0.5 * np.sum([w * (1 - vi * vib) for w, vi, vib in zip(weights, vertices, vertices_b)])

# Distribution de proposition G(theta' | theta_t)
def proposal_distribution(theta_t, epsilon=0.1):
    return theta_t + epsilon * np.random.randn(len(theta_t))

# Distribution d'acceptation A(theta' | theta_t)
def acceptance_distribution(theta_prime, theta_t, vertices, vertices_b, weights, beta=1.0):
    loss_prime = maxcut_objective(vertices, vertices_b, weights)
    loss_t = maxcut_objective(vertices, vertices_b, weights)
    ratio = np.exp(-beta * (loss_prime - loss_t))
    return min(1, ratio)

# Calcul du gradient de la fonction de perte par rapport aux parametres
def calculate_gradient(theta, vertices, vertices_b, weights, epsilon=1e-8):
    gradient = np.zeros_like(theta)
    for i in range(len(theta)):
        theta_plus = theta.copy()
        theta_minus = theta.copy()
        theta_plus[i] += epsilon
        theta_minus[i] -= epsilon

        loss_plus = maxcut_objective(vertices, vertices_b, weights)
        loss_minus = maxcut_objective(vertices, vertices_b, weights)

        gradient[i] = (loss_plus - loss_minus) / (2 * epsilon)

    return gradient

# Algorithme MCMC-VQA avec visualisation de la trajectoire
def mcmc_vqa_with_trajectory(num_episodes_mcmc, num_episodes_vqa, vertices, vertices_b, weights, beta=1.0, epsilon=0.1):
    theta_t = np.random.rand(len(vertices))  # Initialisation al atoire des param tres quantiques

    # Liste pour stocker les trajectoires des param tres
    theta_trajectory = [theta_t.copy()]

```

```
    # Etapes MCMC
    for episode in range(num_episodes_mcmc):
        theta_prime = proposal_distribution(theta_t, epsilon)
        acceptance_prob = acceptance_distribution(theta_prime, theta_t,
    vertices, vertices_b, weights, beta)

        if np.random.rand() < acceptance_prob:
            theta_t = theta_prime

        # Ajouter les param tres    la trajectoire
        theta_trajectory.append(theta_t.copy())

    # Initialisation avec les meilleurs param tres trouv s pendant
   MCMC
    theta_min = theta_t

    # Etapes de VQA pour convergence rapide
    for episode in range(num_episodes_vqa):
        gradient = calculate_gradient(theta_t, vertices, vertices_b,
   weights)
        learning_rate = 0.1  # R glage du taux d'apprentissage
        theta_t = theta_t - learning_rate * gradient

        # Ajouter les param tres    la trajectoire
        theta_trajectory.append(theta_t.copy())

    return theta_min, np.array(theta_trajectory)

# Exemple d'utilisation avec visualisation de la trajectoire
num_vertices = 4
edges = [(0, 1), (1, 2), (2, 3), (3, 0)]
weights = [0.5, 0.8, 0.6, 0.7]
vertices = [1, -1, 1, -1]
vertices_b = [-1, 1, -1, 1]

num_episodes_mcmc = 1000
num_episodes_vqa = 500

result_theta_min, trajectory = mcmc_vqa_with_trajectory(
    num_episodes_mcmc, num_episodes_vqa, vertices, vertices_b, weights)

# Visualisation de la trajectoire
plt.figure(figsize=(10, 6))
for i in range(len(vertices)):
    plt.plot(trajectory[:, i], label=f'$\\theta_{i}$')

plt.xlabel(' tapes  MCMC et VQA')
plt.ylabel('Valeur des param tres')
plt.legend()
plt.title('Trajectoire des param tres  g  n  r  s par MCMC-VQA')
plt.show()
```

Ce qui donne :

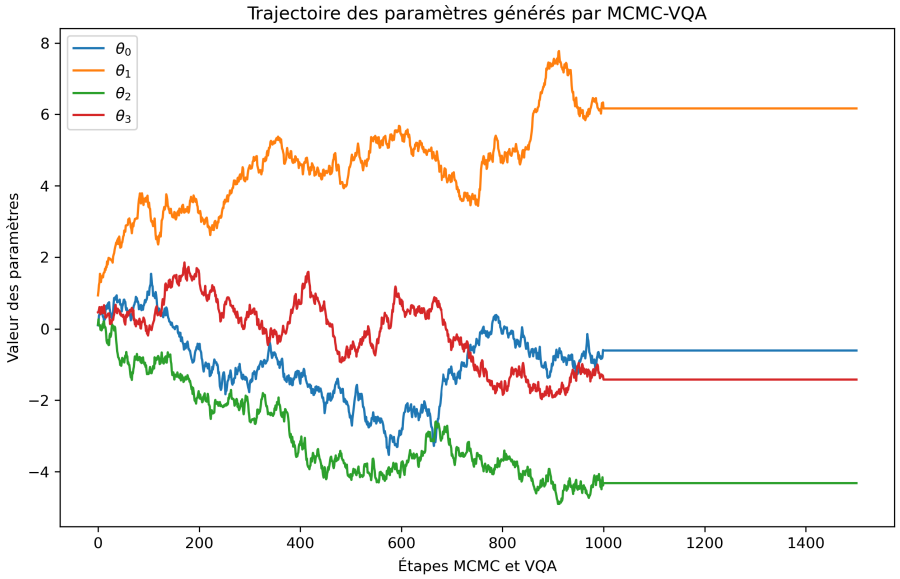



Figure 3: Trajectoire des paramètres générés par MCMC-VQA

# V Preuve d'ergodicité

Si un algorithme Metropolis-Hastings est irréductible et apériodique, alors la chaîne de Markov résultante est de manière provable ergodique. Autrement dit, elle explorera toutes les zones de la distribution de probabilité, convergeant en moyenne vers la distribution stationnaire unique du processus markovien, qui inclut le minimum global de l'espace des solutions. De plus, comme nous avons choisi d'échantillonner à partir de la distribution de Boltzmann de la fonction de perte, nous échantillonnons à partir d'états près des solutions optimales avec une probabilité exponentiellement plus élevée.

# VI Conclusion

L'algorithme MCMC-VQA présente une approche novatrice pour résoudre le problème MaxCut en combinant les techniques de chaînes de Markov Monte Carlo et d'optimisation quantique variationnelle. En exploitant les avantages de chaque méthode, MCMC-VQA parvient à surmonter les obstacles des minima locaux et à converger rapidement vers des solutions de haute qualité. Cette approche hybride offre une nouvelle perspective sur la résolution de problèmes d'optimisation difficile en utilisant des techniques quantiques et classiques de manière synergique.