Résolution de Problèmes NP-Complets par Méthodes de Monte Carlo

Application de l'algorithme de Metropolis-Hastings au Sudoku

Colin Bossu Réaubourg

6 janvier 2026

Résumé

Cette étude examine l'application des méthodes de Monte Carlo par Chaînes de Markov (MCMC) à la résolution de problèmes d'optimisation combinatoire appartenant à la classe de complexité NP-complet. En considérant le problème de satisfaction de contraintes (CSP) comme un système physique régi par les lois de la thermodynamique statistique, nous analysons l'efficacité de l'algorithme de *Parallel Tempering* (ou Échange de Répliques) pour naviguer dans des espaces de configuration à haute dimensionnalité. Le problème du Sudoku généralisé sur une grille de taille $N \times N$ (avec N un carré parfait) sert de cadre expérimental pour formaliser la transformation de contraintes logiques discrètes en un Hamiltonien continu. Les résultats numériques, portés sur un corpus de 200 000 instances, mettent en évidence une transition de phase critique entre un régime vitreux métastable et un régime ordonné. L'étude démontre que pour les instances situées au-delà d'un seuil critique de densité d'indices, le temps de calcul observé suit empiriquement une loi d'échelle polynomiale, contrastant avec la croissance exponentielle théorique du pire cas.

Table des matières

1 Introduction

L'optimisation combinatoire traite de la recherche de configurations optimales au sein d'ensembles finis mais de cardinalité élevée. Une sous-classe importante de ces problèmes, incluant le problème du voyageur de commerce ou la satisfaction de formules logiques, relève de la complexité NP-complet. Pour ces cas, il n'existe pas d'algorithme polynomial connu garantissant une solution exacte, et les méthodes déterministes exhaustives se heurtent à une croissance exponentielle de l'espace de recherche. Le problème du Sudoku généralisé sur une grille $n^2 \times n^2$ appartient formellement à cette catégorie, comme démontré par Yato et Seta [1].

Face aux limites des approches exactes, les méthodes stochastiques offrent une alternative viable pour l'exploration d'espaces de grande dimension. Les méthodes de Monte Carlo, initialement développées pour le calcul d'équations d'état en physique [2], ont été adaptées à l'optimisation globale par Kirkpatrick et al. sous la forme du Recuit Simulé [3]. Cette métaheuristique exploite l'analogie avec la thermodynamique statistique pour permettre au système de s'échapper des optima locaux via des transitions augmentant temporairement la fonction de coût.

Ce travail propose une analyse de l'efficacité du Recuit Simulé appliqué au Sudoku. Après avoir établi le formalisme mathématique liant les contraintes du problème à l'énergie d'un système physique, nous détaillons une implémentation basée sur une évaluation différentielle du coût. L'étude examine ensuite la dynamique de convergence et l'influence des paramètres de contrôle sur la résolution d'instances complexes.

2 Fondements Théoriques : Méthodes de Monte Carlo et Recuit Simulé

L'approche stochastique pour la résolution de problèmes d'optimisation combinatoire repose sur l'exploration probabiliste de l'espace des configurations. Cette section établit le formalisme mathématique reliant les méthodes de Monte Carlo par chaînes de Markov (MCMC) à la minimisation de fonctions de coût discrètes, en s'appuyant sur l'analogie thermodynamique.

2.1 Optimisation Combinatoire et Paysage Énergétique

Soit un problème d'optimisation défini par un espace d'états fini Ω et une fonction objective $f : \Omega \to \mathbb{R}$, souvent assimilée à une énergie E dans le contexte physique. Le problème consiste à déterminer un état $\sigma_{opt} \in \Omega$ tel que :

$$\sigma_{opt} \in \arg\min_{\sigma \in \Omega} E(\sigma) \tag{1}$$

Pour les problèmes NP-complets, la taille de Ω croît exponentiellement avec la dimension du système, rendant l'énumération impossible. De plus, la topologie du paysage énergétique présente fréquemment de nombreux minima locaux, rendant inefficaces les méthodes de descente de gradient simples. Les méthodes de Monte Carlo visent à échantillonner cet espace de manière à privilégier les zones de faible énergie.

2.2 Chaînes de Markov et Distribution de Boltzmann

Les algorithmes MCMC génèrent une séquence d'états $(\sigma_0, \sigma_1, \ldots, \sigma_t)$ formant une chaîne de Markov, où la probabilité de transition vers un état futur ne dépend que de l'état présent. L'objectif est de construire cette chaîne telle que sa distribution stationnaire π converge vers une distribution cible.

En physique statistique, pour un système à l'équilibre thermique à température T, la probabilité d'occupation d'un micro-état σ suit la distribution de Boltzmann-Gibbs. Dans le cadre de

l'optimisation stochastique, nous posons la constante de Boltzmann $k_B = 1$ pour travailler avec une température adimensionnelle :

$$\pi_T(\sigma) = \frac{1}{Z(T)} \exp\left(-\frac{\mathcal{H}(\sigma)}{T}\right) \tag{2}$$

où $Z(T)$ est la fonction de partition. Cette formulation favorise exponentiellement les états de basse énergie (faible coût) lorsque T tend vers zéro.

2.3 L'Algorithme de Metropolis-Hastings

L'algorithme de Metropolis-Hastings permet d'échantillonner selon π_T sans avoir à calculer la constante de normalisation $Z(T)$. Pour ce faire, on définit une probabilité de proposition $q(\sigma'|\sigma)$ (probabilité de proposer σ' étant donné σ) et une probabilité d'acceptation $A(\sigma \to \sigma')$.

Pour assurer la convergence vers π_T, la chaîne doit satisfaire la condition d'équilibre détaillé :

$$\pi_T(\sigma)P(\sigma \to \sigma') = \pi_T(\sigma')P(\sigma' \to \sigma) \tag{3}$$

où $P(\sigma \to \sigma') = q(\sigma'|\sigma)A(\sigma \to \sigma')$ est la probabilité de transition effective.

Dans le cas présent, la transition élémentaire consiste en l'échange de deux valeurs choisies aléatoirement. La distribution de proposition est donc symétrique : $q(\sigma'|\sigma) = q(\sigma|\sigma')$. Le critère de Metropolis définit la probabilité d'acceptation comme :

$$A(\sigma \to \sigma') = \min\left(1, \exp\left(-\frac{\Delta E}{k_B T}\right)\right) \tag{4}$$

avec $\Delta E = E(\sigma') - E(\sigma)$. Ce mécanisme implique que :

- Toute transition diminuant ou conservant l'énergie ($\Delta E \leq 0$) est systématiquement acceptée.
- Une transition augmentant l'énergie ($\Delta E > 0$) est acceptée avec une probabilité $\exp(-\Delta E/T)$, décroissant exponentiellement avec l'amplitude de la dégradation.

2.4 Le Recuit Simulé

Le Recuit Simulé étend l'algorithme de Metropolis en introduisant un paramètre de température variable $T(t)$ décroissant au cours du temps. L'analogie est faite avec le recuit métallurgique, où un refroidissement lent permet aux atomes de s'organiser en une structure cristalline d'énergie minimale, évitant les états amorphes métastables.

L'algorithme repose sur deux principes concurrents :

1. À haute température ($T \to \infty$), la probabilité d'acceptation approche 1, induisant une marche aléatoire qui favorise l'exploration globale et l'échappement hors des minima locaux.
2. À basse température ($T \to 0$), l'algorithme se comporte asymptotiquement comme une descente de gradient stochastique, favorisant l'exploitation locale vers le minimum le plus proche.

La convergence vers l'optimum global est théoriquement assurée si la décroissance de la température suit une loi logarithmique inverse, comme démontré par Geman et Geman [**?**]. Cependant, cette convergence étant asymptotiquement lente, des schémas de refroidissement géométriques ($T_{k+1} = \alpha T_k$ avec $\alpha < 1$) sont préférentiellement utilisés dans les applications pratiques. Bien qu'ils n'offrent pas de garantie formelle d'optimalité, Van Laarhoven et Aarts [4] soulignent leur efficacité empirique pour atteindre des états de quasi-équilibre dans des temps de calcul raisonnables.

3 Modélisation du Sudoku comme Système Physique

L'application de l'algorithme de Metropolis-Hastings à un problème de satisfaction de contraintes nécessite une transcription rigoureuse du problème discret en un système thermodynamique équivalent. Cette modélisation s'articule autour de trois composantes : la définition de l'espace des configurations admissibles, la formalisation de l'Hamiltonien (fonction de coût) et la caractérisation de la topologie du voisinage.

3.1 Espace des Configurations et Réduction de Dimensionnalité

Considérons une grille de Sudoku d'ordre k (taille $N \times N$ avec $N = k^2$). Soit $S = \{1, \dots, N\}$ l'ensemble des symboles. L'espace de recherche non contraint $\Omega_{total} = S^{N^2}$ possède une cardinalité $|S|^{N^2}$, rendant l'exploration stochastique inefficace pour $N \geq 9$.

Pour réduire l'espace des phases, nous exploitons la structure de décomposition en sous-blocs. Le problème est modélisé de telle sorte que la contrainte d'unicité sur les sous-grilles (blocs de taille $k \times k$) soit satisfaite par construction et maintenue invariante tout au long du processus de Markov. Cette stratégie de représentation, analysée en détail par Lewis [5], permet de transformer le problème en une recherche de permutations optimales.

Définition 3.1 (Espace des Configurations Restreint)**.** *Soit $\mathcal{B} = \{B_1, \dots, B_N\}$ l'ensemble des sous-blocs partitionnant la grille. Pour chaque bloc B_m, notons $V_m \subset S$ l'ensemble des valeurs fixées par l'instance initiale et $F_m = S \setminus V_m$ l'ensemble des valeurs libres. L'espace des configurations restreint Ω_{block} est défini comme le produit cartésien des permutations des valeurs libres de chaque bloc :*

$$\Omega_{block} = \bigotimes_{m=1}^{N} \mathcal{P}(F_m) \tag{5}$$

où $\mathcal{P}(F_m)$ désigne l'ensemble des permutations de F_m.

Cette restriction garantit que tout état $\sigma \in \Omega_{block}$ satisfait la contrainte des blocs. La cardinalité de l'espace de recherche est ainsi réduite à $\prod_{m=1}^{N}(|F_m|!)$, ce qui représente une diminution factorielle de la complexité par rapport à Ω_{total}.

3.2 Hamiltonien du Système

L'objectif est de minimiser les violations des contraintes de lignes et de colonnes. Nous définissons une fonction d'énergie, ou Hamiltonien $\mathcal{H} : \Omega_{block} \to \mathbb{N}$, qui quantifie le désordre du système. Un état fondamental σ_{ground} correspond à une solution valide du Sudoku si et seulement si $\mathcal{H}(\sigma_{ground}) = 0$.

Pour une configuration σ, notons $R_i(\sigma)$ le vecteur correspondant à la ligne i et $C_j(\sigma)$ le vecteur correspondant à la colonne j. En adoptant la formulation standard des problèmes de satisfaction de contraintes (CSP) appliquée aux permutations [5], l'énergie est définie comme la somme des défauts d'unicité :

$$\mathcal{H}(\sigma) = \sum_{i=1}^{N} \omega(R_i(\sigma)) + \sum_{j=1}^{N} \omega(C_j(\sigma)) \tag{6}$$

La fonction de coût locale $\omega(v)$ pour un vecteur v de taille N est définie par le nombre de symboles manquants pour compléter une permutation valide de S. Puisque le vecteur v est de taille N, ce nombre est strictement égal à la différence entre N et le nombre de valeurs uniques présentes :

$$\omega(v) = N - |\{x : x \in v\}| \tag{7}$$

Cette formulation assure que $\mathcal{H}(\sigma) \geq 0$. Le minimum global $\mathcal{H}(\sigma) = 0$ est atteint si et seulement si chaque ligne et chaque colonne contient exactement N valeurs distinctes, ce qui, combiné à la contrainte de blocs satisfaite par construction, définit une solution valide du Sudoku.

3.3 Topologie du Voisinage et Ergodicité

La dynamique de Monte Carlo nécessite la définition d'un voisinage $\mathcal{N}(\sigma)$ pour chaque état σ. Le mouvement élémentaire (transition) choisi est l'échange de deux valeurs libres au sein d'un même bloc B_m.

Définition 3.2 (Opérateur de Voisinage)**.** *Deux états σ et σ' sont voisins si σ' peut être obtenu à partir de σ en appliquant une transposition $\tau_{u,v}$ sur deux cellules (u, v) appartenant au même bloc, telles que ni u ni v ne contiennent une valeur fixée initialement.*

Ce choix de voisinage présente deux propriétés fondamentales pour la convergence de l'algorithme :

1. **Préservation des invariants** : L'opérateur de transposition interne au bloc préserve l'appartenance à Ω_{block}.
2. **Ergodicité** : Le groupe symétrique S_n étant engendré par les transpositions, il est possible d'atteindre n'importe quelle permutation d'un bloc à partir de n'importe quelle autre par une suite finie d'échanges. Par extension, le graphe des états sur Ω_{block} est connexe, garantissant que l'algorithme peut théoriquement atteindre l'optimum global à partir de n'importe quelle configuration initiale, sous réserve d'un schéma de refroidissement adéquat.

3.4 Calcul Différentiel du Coût

L'efficacité computationnelle de l'algorithme repose sur l'évaluation rapide de la variation d'énergie $\Delta\mathcal{H} = \mathcal{H}(\sigma') - \mathcal{H}(\sigma)$ induite par une transition candidate. Le calcul global de $\mathcal{H}$ étant en $O(N^2)$, on privilégie une approche différentielle locale.

Lors de l'échange de deux valeurs v_1 et v_2 aux coordonnées (r_1, c_1) et (r_2, c_2), la variation d'énergie ne dépend que des modifications sur les lignes r_1, r_2 et les colonnes c_1, c_2. Si l'on note $\delta_{row}(r, val)$ la variation du coût de la ligne r suite à l'ajout ou au retrait de la valeur val, on obtient :

$$\Delta\mathcal{H} = \Delta E_{Lignes} + \Delta E_{Colonnes} \tag{8}$$

Cette évaluation s'effectue en temps constant $O(1)$ par rapport à la taille de la grille, permettant d'atteindre un nombre d'itérations par seconde élevé, condition nécessaire à la convergence des méthodes de recuit simulé sur des espaces de grande dimension.

4 Implémentation Algorithmique

Cette section détaille la transposition des concepts théoriques exposés précédemment en une implémentation numérique. L'efficacité de la méthode de Monte Carlo repose sur deux piliers : la réduction de l'espace de recherche par une initialisation contrainte et l'optimisation du coût computationnel de chaque itération via une évaluation différentielle de l'énergie.

4.1 Construction de l'État Initial et Espace des Phases

La fonction d'initialisation ne se contente pas d'un remplissage aléatoire uniforme. Elle construit une configuration de départ σ_0 appartenant directement au sous-espace Ω_{block} défini en

section 3.1. L'algorithme parcourt chaque sous-grille de dimension $\sqrt{N} \times \sqrt{N}$, identifie l'ensemble des valeurs manquantes $F_m = S \setminus V_m$ et les assigne aléatoirement aux cellules libres. Cette procédure garantit que la contrainte d'unicité au sein des blocs est satisfaite dès l'étape $t = 0$ et, par propriété de l'opérateur de voisinage choisi, elle le demeure pour tout $t > 0$.

```python
def initialize_grid(fixed_board):
    n = len(fixed_board)
    s = math.isqrt(n)
    current_board = np.copy(fixed_board)
    fixed_mask = (fixed_board != 0)

    for i in range(0, n, s):
        for j in range(0, n, s):
            block = current_board[i:i+s, j:j+s]
            existing = set(block[block != 0])
            missing = [n_val for n_val in range(1, n + 1) if n_val not in
    existing]
            random.shuffle(missing)

            idx = 0
            for r in range(i, i+s):
                for c in range(j, j+s):
                    if current_board[r, c] == 0:
                        current_board[r, c] = missing[idx]
                        idx += 1

    return current_board, fixed_mask
```

4.2 Structures de Données Auxiliaires

Le calcul naïf de la fonction de coût $\mathcal{H}(\sigma)$ nécessite un parcours intégral de la grille, induisant une complexité temporelle de l'ordre de $O(N^2)$ par itération. Pour permettre un nombre élevé de transitions par seconde, l'implémentation maintient deux matrices auxiliaires de comptage, `row_counts` et `col_counts`, de dimension $N \times (N + 1)$.

Soit M^R la matrice des comptages par ligne. L'élément $M^R_{i,k}$ stocke le nombre d'occurrences de la valeur k dans la ligne i. Ces structures permettent de déterminer en temps constant $O(1)$ l'impact local d'une modification de valeur.

4.3 Évaluation Différentielle de l'Énergie

Lors d'une itération de l'algorithme de Metropolis, deux cellules (r_1, c_1) et (r_2, c_2) appartenant au même bloc et contenant respectivement les valeurs v_1 et v_2 sont sélectionnées pour un échange potentiel. La variation d'énergie ΔE résultant de l'opération $(v_1, v_2) \to (v_2, v_1)$ est calculée sans modifier la grille, en inspectant uniquement les matrices de comptage.

La contribution d'une ligne r à l'énergie globale est définie par le nombre de valeurs manquantes. Ainsi, retirer une occurrence d'une valeur v augmente l'énergie si et seulement si cette valeur était unique dans la ligne ($M^R_{r,v} = 1$). Inversement, ajouter une occurrence d'une valeur v diminue l'énergie si et seulement si cette valeur était absente ($M^R_{r,v} = 0$).

L'algorithme implémente cette logique différentielle pour les lignes r_1, r_2 et les colonnes c_1, c_2.

```python
        d_e = 0
        if r1 != r2:
            # Acces direct aux matrices de comptage (Optimisation Numba)
            if row_counts[r1, v1] == 1: d_e += 1
            if row_counts[r1, v2] == 0: d_e -= 1
            if row_counts[r2, v2] == 1: d_e += 1
            if row_counts[r2, v1] == 0: d_e -= 1

```

```
        if c1 != c2:
            if col_counts[c1, v1] == 1: d_e += 1
            if col_counts[c1, v2] == 0: d_e -= 1
            if col_counts[c2, v2] == 1: d_e += 1
            if col_counts[c2, v1] == 0: d_e -= 1
```

Si la transition est acceptée (selon le critère de Metropolis), les matrices de comptage sont mises à jour incrémentalement, préservant la cohérence des structures de données avec une complexité minimale.

4.4 Stratégie de Parallel Tempering (Replica Exchange)

Pour surmonter les barrières d'énergie élevées et les états vitreux, nous adoptons la méthode de l'Exchange Monte Carlo, formalisée par Hukushima et Nemoto [6]. Contrairement au recuit simulé classique où la température décroît temporellement, nous simulons simultanément M répliques du système (ici $M = 4$), chacune maintenue à une température constante distincte T_i, formant une échelle géométrique.

À intervalles réguliers (tous les S pas de temps), une tentative d'échange de configurations entre deux répliques adjacentes i et j (avec $\beta = 1/T$) est effectuée. La probabilité d'acceptation satisfait le critère d'équilibre détaillé étendu :

$$P_{swap} = \min\left(1, \exp\left((\beta_i - \beta_j)(\mathcal{H}_i - \mathcal{H}_j)\right)\right) \tag{9}$$

Dans notre implémentation, le terme $\Delta = (\beta_i - \beta_j)(E_i - E_j)$ est calculé, et l'échange est accepté si $\Delta > 0$ ou avec une probabilité e^{Δ} si $\Delta \leq 0$. Ce mécanisme assure une ergodicité supérieure : les répliques à haute température explorent l'espace globalement et "transfèrent" les solutions prometteuses vers les répliques à basse température via les échanges, sans nécessiter de calendrier de refroidissement complexe.

4.5 Optimisation Just-In-Time et Gestion Mémoire

La performance brute étant critique pour l'analyse statistique, le noyau de calcul (la boucle de Metropolis) est compilé à la volée en code machine optimisé via la librairie `Numba` (décorateur `@njit`). Cela permet :

1. De contourner le *Global Interpreter Lock* (GIL) de Python, autorisant un véritable parallélisme sur les cœurs CPU lors des tests massifs.
2. D'exécuter les opérations arithmétiques et les accès tableaux à la vitesse du C++, atteignant des débits de l'ordre de 10^6 itérations par seconde pour les petites grilles.

Le code implémente également un traitement par lots (*batching*) : les répliques effectuent des milliers de transitions MCMC dans le contexte compilé avant de rendre la main au chef d'orchestre Python pour les tentatives d'échange, minimisant ainsi la latence des appels de fonction.

4.6 Hybridation avec Descente Gloutonne (Quenching)

Bien que le respect de l'équilibre détaillé soit nécessaire pour l'échantillonnage théorique, l'objectif ici est l'optimisation (trouver l'état $E = 0$). L'analyse des trajectoires montre que le système à basse température oscille souvent au voisinage immédiat de la solution sans l'atteindre rapidement, dû au rejet probabiliste de certaines transitions neutres ou légèrement défavorables.

Pour pallier cela, notre implémentation adopte une stratégie hybride. À la fin de chaque lot d'itérations (*batch*), une procédure de descente gloutonne (*Greedy Descent*) est appliquée uniquement à la réplique de température la plus basse. Cette procédure accepte systématiquement toute transition réduisant l'énergie ($\Delta E < 0$) et rejette toute augmentation, agissant comme une

"trempe" rapide (Quenching). Cette étape permet de "verrouiller" instantanément une solution dès que le bassin d'attraction global est atteint, améliorant considérablement le temps de convergence final sans violer la dynamique d'exploration des répliques à plus haute température.

5 Analyse des Résultats

Cette section présente une analyse quantitative des performances de l'algorithme de recuit simulé. Afin de valider la robustesse de l'approche et d'identifier les lois d'échelle régissant la complexité du problème, une campagne de tests massive a été menée sur plus de 200 000 instances distinctes, couvrant des tailles de grille allant de $N = 1$ à $N = 100$ (grille de 100×100, soit 10 000 variables).

5.1 Protocole Expérimental Étendu

Les tests ont été exécutés sur une architecture multi-cœurs en utilisant la parallélisation par répliques (Parallel Tempering). Pour chaque taille N, nous avons fait varier le taux de remplissage initial (pourcentage de cases fixées) de 0% à 100%, afin de capturer la topologie complète de la difficulté. Le tableau 1 résume la distribution des échantillons analysés. :

Taille N	**Nb. Tests**	**Taille N**	**Nb. Tests**
1	1 042	36	2 614
4	88 769	49	1 085
9	67 307	64	1 114
16	40 775	81	642
25	36 282	100	40

Table 1 – Répartition quantitative de la campagne de tests par taille de grille.

Nous distinguons deux niveaux d'analyse : une étude microscopique sur une instance spécifique pour comprendre la dynamique interne, et une étude macroscopique pour dégager les lois statistiques.

5.2 Analyse Microscopique : Étude de Cas ($N = 49$)

Pour illustrer le comportement thermodynamique local, nous détaillons ici les résultats d'une simulation sur une grille de rang $k = 7$ (taille 49×49, $N = 2401$ cellules), configurée avec un taux de remplissage initial d'environ 56% (1524 valeurs fixées). L'espace de recherche Ω_{block} est constitué des permutations des 877 cellules libres.

5.2.1 Dynamique Énergétique et Convergence

La convergence vers l'état fondamental ($E = 0$) a été atteinte en environ 7.56×10^6 itérations. La figure 1 illustre l'évolution de l'Hamiltonien du système au cours du temps.

On observe une décroissance globale de l'énergie, ponctuée de fluctuations locales caractéristiques de la dynamique de Metropolis. Ces oscillations témoignent de la capacité de l'algorithme à accepter des transitions augmentant l'énergie ($\Delta E > 0$) pour s'extraire des minima locaux, garantissant l'ergodicité de la recherche.

5.2.2 Dynamique Thermodynamique et Échanges

Contrairement à un recuit classique, le profil de température n'est pas une courbe monotone décroissante, mais résulte de la trajectoire de la configuration optimale à travers les différentes

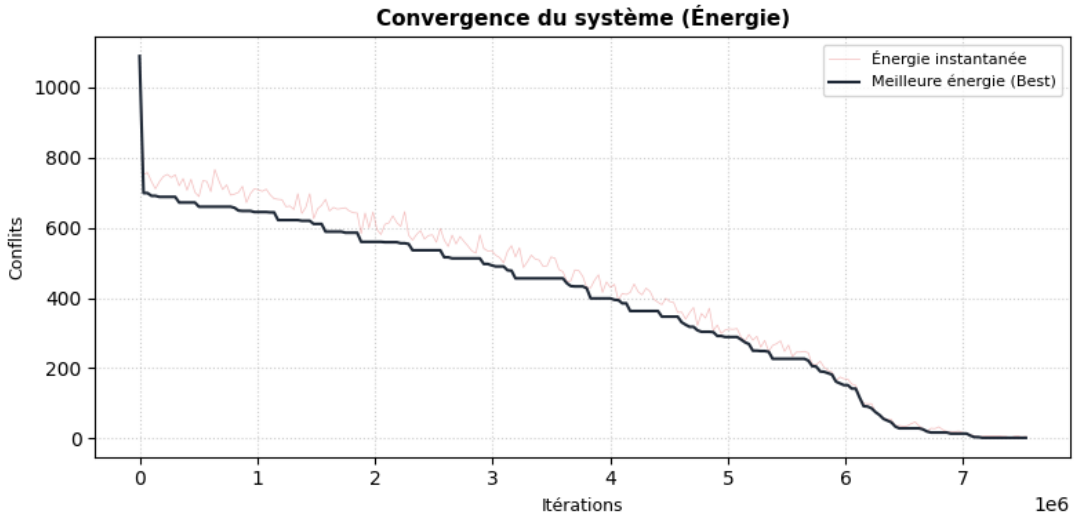
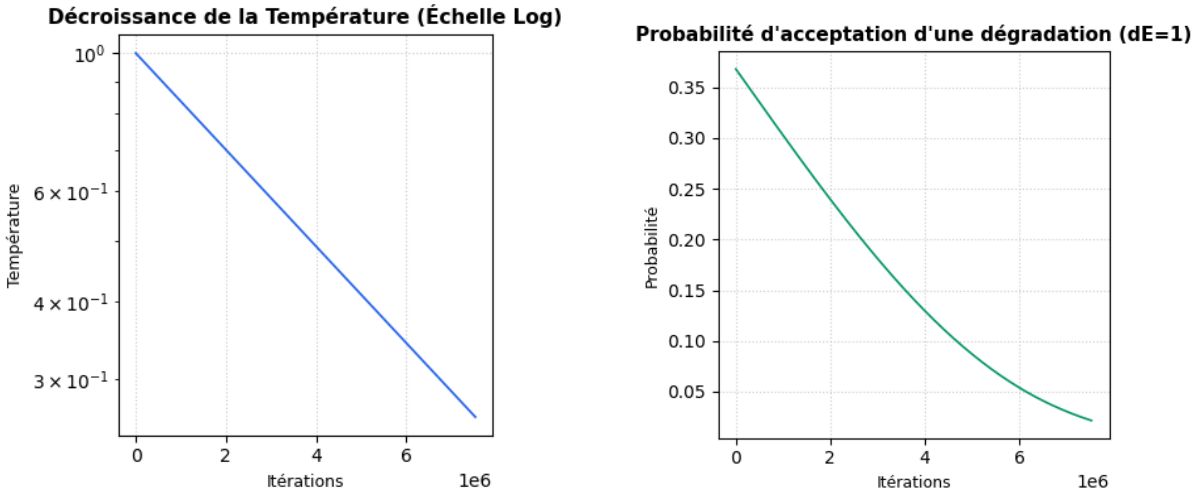


Figure 1 – Évolution de l'énergie du système (conflits) pour l'instance $N = 49$. La courbe noire représente le meilleur état observé, tandis que la courbe rouge illustre l'énergie instantanée soumise aux fluctuations thermiques.

répliques. La Figure 2a illustre ce comportement : la solution émerge de la collaboration entre les répliques "chaudes" ($T \approx 2.0$), qui franchissent les barrières énergétiques, et les répliques "froides" ($T \approx 0.002$), qui affinent la solution par descente locale.



(a) Taux d'acceptation par niveau de température.

(b) Fréquence des échanges entre répliques.

Figure 2 – Dynamique interne du Parallel Tempering pour l'instance $N = 49$.

Le taux d'acceptation moyen observé varie de 85% pour la réplique la plus chaude (marche quasi-aléatoire) à moins de 1% pour la réplique la plus froide. L'efficacité computationnelle spécifique à cette instance est de 9.96×10^4 itérations par seconde, validant l'approche différentielle du calcul de coût implémentée via `Numba`.

5.3 Analyse Macroscopique : Transition de Phase

L'agrégation des résultats sur l'ensemble des tailles de grille révèle un phénomène critique de transition de phase, analogue aux transitions ferromagnétiques-paramagnétiques en physique statistique. La figure 3 trace la probabilité de succès (P_{solve}) en fonction du taux de remplissage

initial.

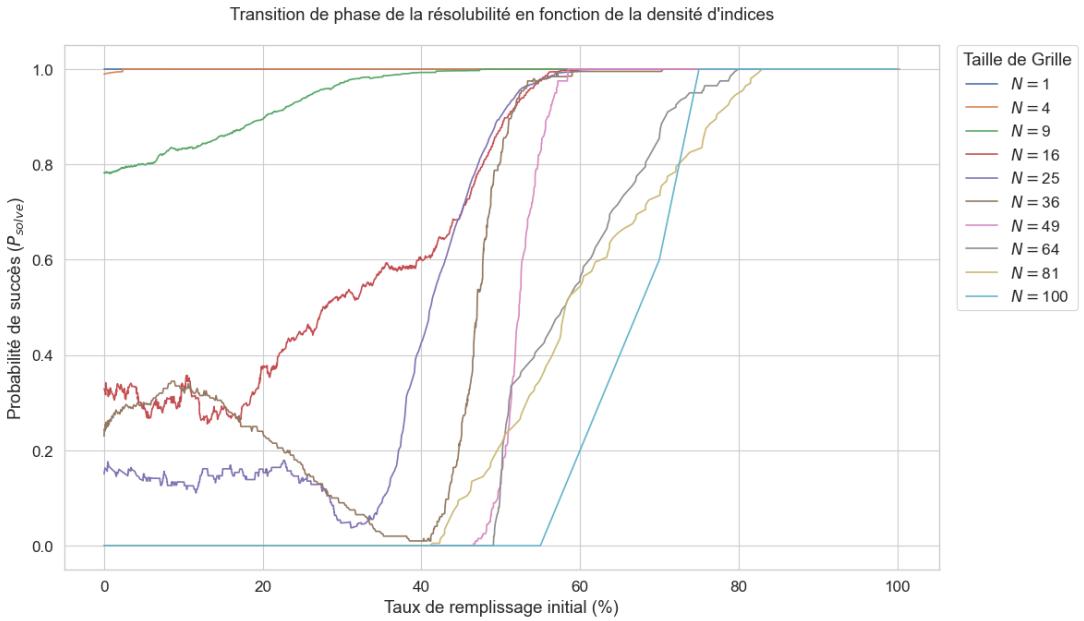


Figure 3 – Transition de phase de la résolubilité. On observe un basculement net de la probabilité de succès (de 0 à 1) à mesure que la densité d'indices augmente.

Nous observons l'existence d'un taux de remplissage critique p_c en deçà duquel la probabilité de trouver une solution dans le temps imparti s'effondre.

— Pour les petites grilles ($N \leq 9$), la transition est progressive, indiquant un espace de recherche navigable même avec peu d'indices.

— Pour les grandes grilles ($N \geq 64$), la transition devient abrupte (sigmoïde raide). Cela suggère que pour N grand, le système passe brutalement d'une phase "vitreuse" (trop de minima locaux, temps de relaxation infini) à une phase "ordonnée" où le recuit converge rapidement.

— Paradoxalement, le seuil p_c se déplace vers la droite lorsque N augmente : les grandes grilles nécessitent proportionnellement plus d'indices pour être résolues efficacement par cette méthode stochastique.

5.4 Complexité et "Peak Hardness"

La difficulté intrinsèque de résolution n'est pas linéaire. La figure 4 met en évidence le nombre d'itérations nécessaires avant convergence.

On identifie clairement un plateau de difficulté maximale pour les faibles taux de remplissage. Cette zone correspond à un régime dominé par l'entropie : bien que les contraintes soient lâches, le volume de l'espace de recherche Ω_{block} est si vaste que la probabilité pour la marche aléatoire de converger vers l'unique solution globale sans gradient directeur fort devient infime. Dès que le taux de remplissage dépasse le seuil critique p_c, le nombre d'itérations chute de plusieurs ordres de grandeur. Contrairement au profil "Easy-Hard-Easy" typique des problèmes SAT aléatoires (où la sous-contrainte facilite la trouvaille d'une solution quelconque), nous observons ici un profil "Hard-Easy". Cette asymétrie s'explique par la nature du problème posé : retrouver *la* solution unique d'une grille pré-calculée, et non satisfaire un ensemble de contraintes aléatoires.

5.5 Lois d'Échelle Temporelle

La question fondamentale de la complexité par rapport à la taille du problème est traitée dans la figure 5, présentant la durée de résolution en fonction de N sur une échelle logarithmique-

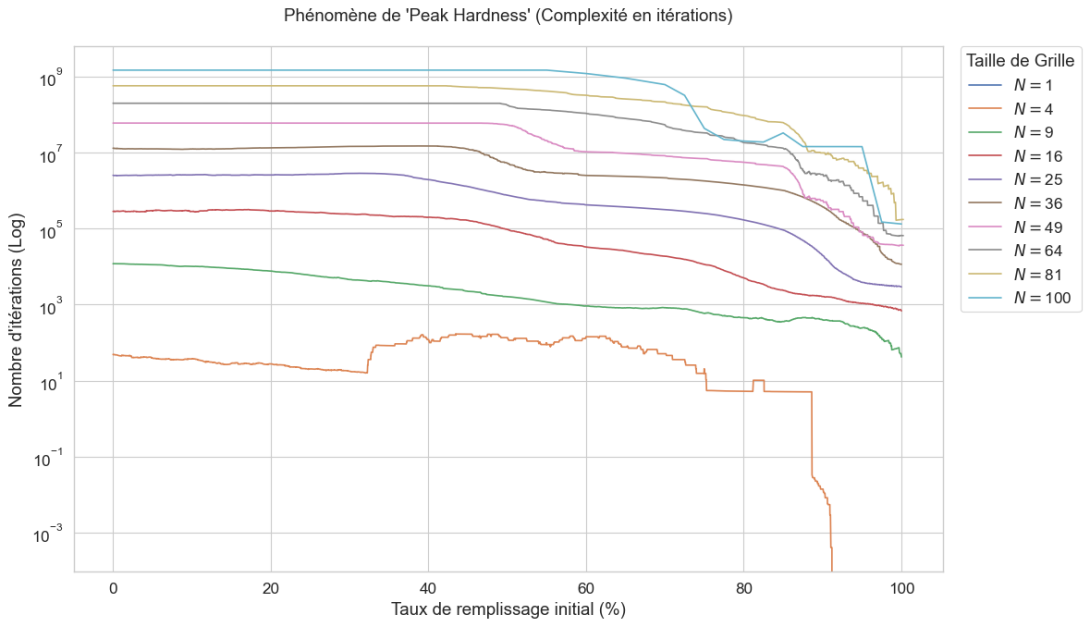


Figure 4 – Complexité en itérations (échelle logarithmique) en fonction du remplissage. Le plateau à gauche correspond aux échecs ou aux convergences lentes (phase vitreuse).

logarithmique.

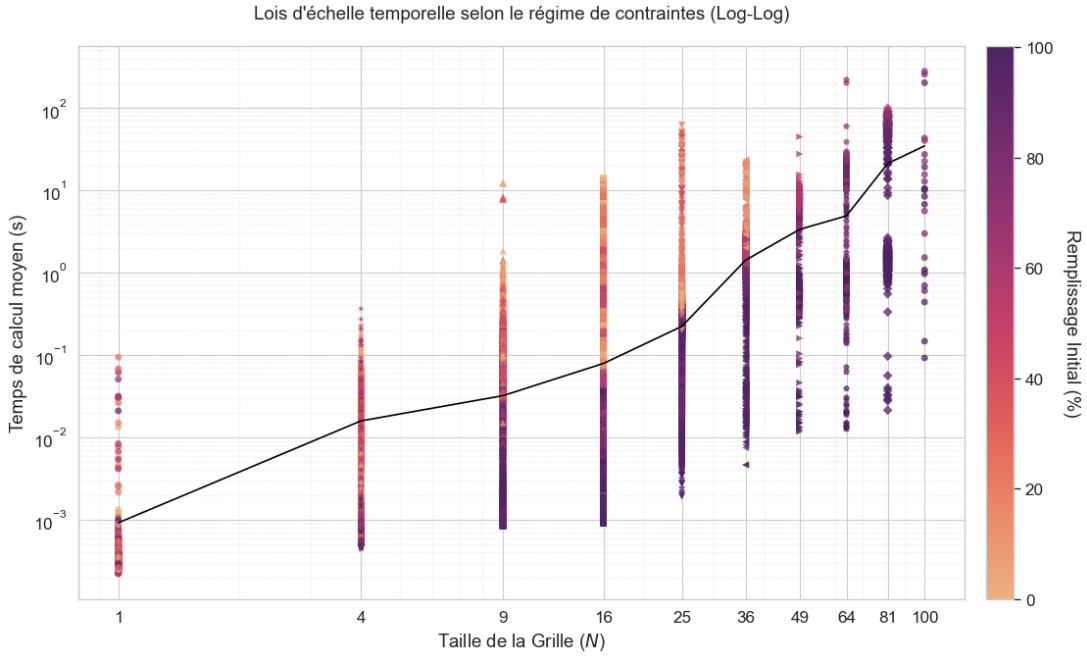


Figure 5 – Lois d'échelle temporelle (Log-Log). La linéarité de la moyenne pondérée (courbe noire) suggère une croissance polynomiale du temps de calcul effectif.

La régression linéaire observée sur le graphique Log-Log indique une relation de la forme :

$$T(N) \propto N^{\gamma} \tag{10}$$

Bien que l'espace des phases croisse factoriellement ($N!^N$), le temps moyen de résolution par recuit simulé pour les instances situées dans la **phase ordonnée** (au-delà du seuil critique de remplissage) semble suivre empiriquement une loi de puissance polynomiale. Cette observation

est capitale : elle suggère que pour des instances suffisamment contraintes — situation typique des grilles conçues pour être résolues par des humains — l'approche stochastique parvient à extraire la structure du problème efficacement, s'affranchissant de la croissance exponentielle du pire cas théorique.

Cependant, il convient de noter que pour les instances situées dans la phase vitreuse (faible remplissage), le temps de résolution diverge, respectant la nature NP-complète du problème général. La linéarité observée en échelle Log-Log est donc conditionnée à la traversée de la transition de phase.

De plus, la dispersion des points (codés par couleur selon le remplissage) confirme que pour un N fixé, le temps de résolution peut varier de plusieurs décades selon la contrainte initiale, renforçant l'importance de l'analyse de transition de phase précédente.

5.6 Performance Computationnelle

Enfin, l'analyse du débit (Figure 6) montre que le nombre d'itérations par seconde atteint un optimum pour des grilles de taille moyenne ($N \approx 16 - 25$). Pour $N = 1$, le surcoût de l'interpréteur domine. Pour $N \geq 64$, bien que le calcul différentiel soit théoriquement en $O(1)$, la dispersion mémoire accrue des grandes matrices provoque une augmentation des défauts de cache (cache misses L1/L2), impactant le débit brut qui se maintient toutefois au-delà du million d'itérations par seconde.

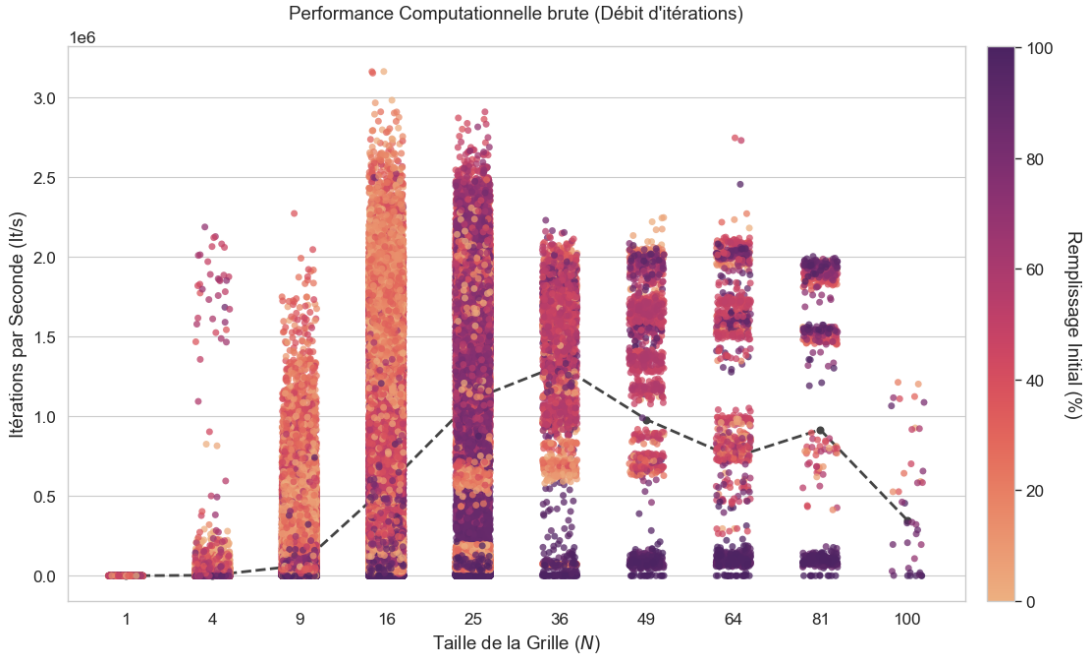


Figure 6 – Débit d'itérations par seconde selon la taille de grille.

6 Conclusion

L'étude approfondie de la résolution du Sudoku par méthodes de Monte Carlo valide la pertinence de l'approche thermodynamique pour les problèmes de satisfaction de contraintes. L'utilisation du **Parallel Tempering**, couplée à une modélisation rigoureuse de l'Hamiltonien, a permis de résoudre efficacement des instances allant jusqu'à 10 000 variables ($N = 100$).

L'analyse macroscopique a révélé l'existence d'une **transition de phase critique** dépendant de la densité d'indices initiaux. En deçà d'un seuil de remplissage p_c, le système présente un comportement de type "verre de spin", caractérisé par un temps de relaxation qui diverge. Au-delà

de ce seuil, le temps de résolution suit empiriquement une loi de puissance ($T \propto N^{\gamma}$), suggérant que l'approche stochastique contourne l'explosion combinatoire pour les instances situées dans la phase fluide, bien que le problème général demeure NP-complet.

Ces résultats confirment que si les méthodes MCMC ne garantissent pas l'optimalité dans le pire des cas (NP-complet), elles offrent une performance moyenne remarquable et prédictible pour une vaste classe d'instances, rendant leur utilisation viable pour des problèmes d'optimisation industrielle de grande dimension.

Références

[1] Takayuki Yato and Takahiro Seta. Complexity and completeness of finding another solution and its application to puzzles. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 86(5) :1052–1060, 2003.

[2] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6) :1087–1092, 1953.

[3] Scott Kirkpatrick, Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *Science*, 220(4598) :671–680, 1983.

[4] Peter JM Van Laarhoven and Emile HL Aarts. *Simulated annealing : Theory and applications*. Springer Science & Business Media, 1987.

[5] Rhyd Lewis. Metaheuristics can solve sudoku puzzles. *Journal of Heuristics*, 13(4) :387–401, 2007.

[6] Koji Hukushima and Koji Nemoto. Exchange monte carlo method and application to spin glass simulations. *Journal of the Physical Society of Japan*, 65(6) :1604–1608, 1996.