
RAPPORT DE MINI-PROJET

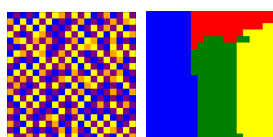
API0074 - MODÉLISATION STOCHASTIQUE - A23

ÉTUDIANTS

SUN HUDIE
LAI JINXING

ENSEIGNANT ENCADRANT

LIMNIOS NIKOLAOS



13 SEPTEMBRE 2024

UNIVERSITÉ DE TECHNOLOGIE DE COMPIÈGNE

Table des matières

1	Configuration	1
2	Séparation - SUN Hudie	2
2.1	Critère d'amélioration : Calcul d'énergie	2
2.2	Algorithme de Metropolis	2
2.3	Conclusion	3
3	Stratification - LAI Jinxing	4
3.1	Réalisation : principe de l'algorithme recuit simulé	4
3.2	Fonction d'énergie	5
3.3	Conclusion	5

1. Configuration

D'après l'énoncé, nous devons placer aléatoirement 50 étudiants de RO05, 100 de SY01, 150 de MT21 et 100 de GE27 dans 400 places disponibles, organisées en 20 rangées de 20 tables chacune. Pour atteindre cet objectif, nous procéderons comme suit : nous commencerons par générer une liste vide représentant les 400 places. Ensuite, nous choisirons aléatoirement une UV parmi les UV restantes, qui sont RO05, SY01, MT21 et GE27, et l'ajouterons à la fin de notre liste de places. Lorsqu'une UV aura été complètement attribuée, nous la retirerons de l'ensemble des UV restantes. Nous continuerons ce processus jusqu'à ce que toutes les places soient attribuées. Finalement, on convertit la liste des place en une matrice de taille 20×20 .

Figure 1 montre la configuration initiale obtenue.

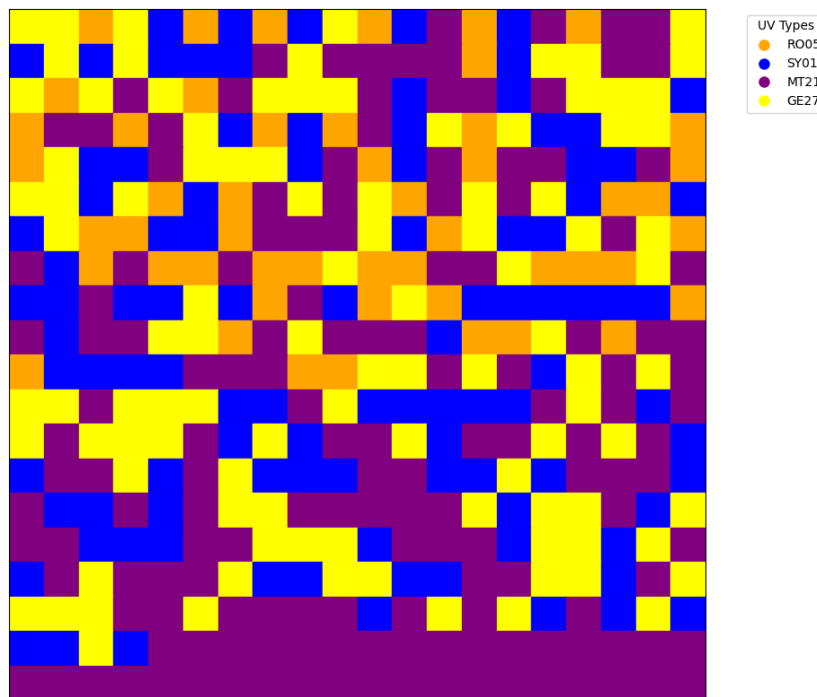


FIGURE 1 – Initialisation des places

2. Séparation - SUN Hudie

2.1 Critère d'amélioration : Calcul d'énergie

Pour appliquer l'algorithme de Metropolis, il est nécessaire de déterminer l'énergie du système.

Dans le cadre de la tâche de séparation, on définit : dans une case $a_{i,j}$, $i, j \in \{0, 19\}$, si son voisin de gauche est de la même UV et ses trois autres voisins sont d'UV différentes, alors l'énergie de cette case est égale à $1 + 0 \times 3 = 1 + 0 \times 3 = 1$. L'énergie totale du système est la somme des énergies de toutes les cases, divisée par 2, car l'énergie de chaque paire de voisins est comptée deux fois.

L'énergie de la position initiale représentée dans la figure 1, que nous avons calculée, est de 259.

2.2 Algorithme de Metropolis

La mission de cette section est d'appliquer l'algorithme de Metropolis dans le but de disposer les étudiants de manière à éviter, autant que possible, que deux étudiants de la même UV se trouvent côte à côte sur des tables voisines.

L'algorithme de Metropolis se déroule comme suit :

1. Calculer l'énergie U_n de l'état actuel du système S_n .
2. Sélectionner aléatoirement deux positions s et s' , effectuer leur échange, puis calculer l'énergie U'_n du système après cet échange.
3. Calculer la différence d'énergie $\Delta U = U'_n - U_n$:
 - Si $\Delta U < 0$, valider l'échange des positions s et s' , conduisant à un nouvel état du système S_{n+1} .
 - Sinon, effectuer l'échange des positions s et s' avec une probabilité de $e^{-\Delta U / temp}$, menant également à S_{n+1} .

Ainsi, l'état S_{n+1} peut être identique à S_n en termes de configuration, ou différent suite à l'échange des positions s et s' .

N.B. : Ici, le terme *temp* désigne la **température**. Il est clair qu'une température élevée peut entraîner une probabilité accrue de permutation des places. La figure 2 illustre l'impact de la température sur le processus de convergence du système.

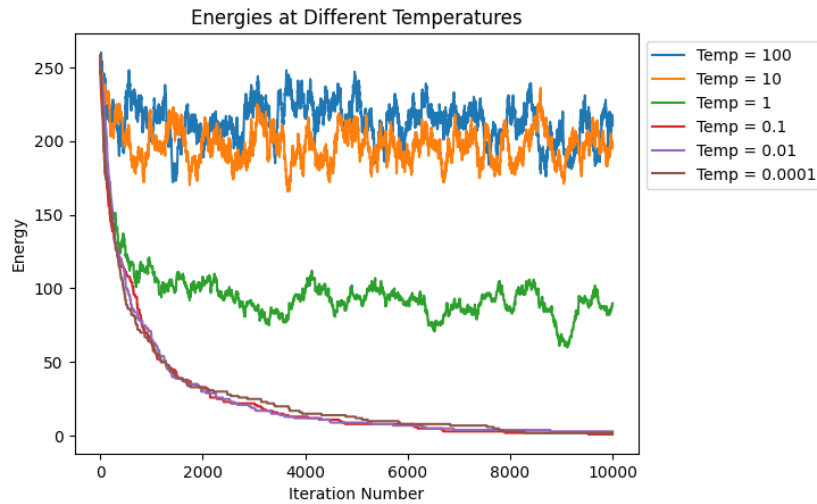


FIGURE 2 – Variation d'énergie en fonction du nbr. d'itérations pour différentes températures

La Figure 3 présente le résultat obtenu après 10 000 itérations, en utilisant une `temperature` fixée à 0.1. Concernant l'énergie du système, elle est évaluée à une valeur de 4.

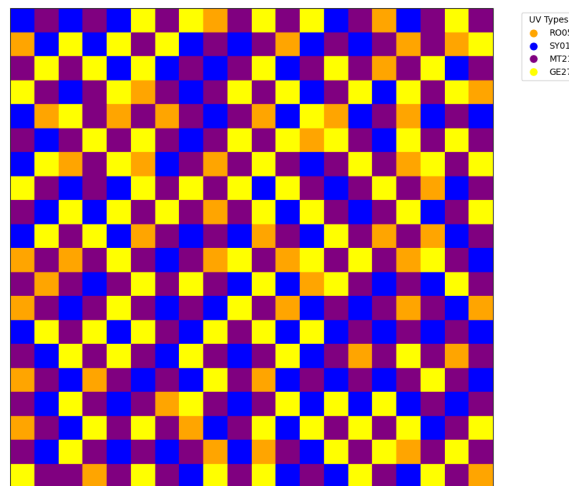


FIGURE 3 – Résultat obtenu avec `temperature=0.1` et `iteration=10000`

2.3 Conclusion

On peut observer que lorsque la température est très élevée, le système est extrêmement instable car nous échangeons très fréquemment deux positions quelconques, indépendamment du fait que cet échange puisse conduire à une énergie totale plus faible. Cela permet de sortir des solutions locales optimales pour trouver la solution globale optimale. Cependant, étant donné que le problème à résoudre est relativement simple dans notre cas, l'utilisation d'une température très basse, c'est-à-dire en effectuant des échanges de positions uniquement lorsque $\Delta < 0$, permet également d'atteindre la solution optimale en 10 000 cycles. Pour des problèmes plus complexes, je pense qu'il serait judicieux de commencer avec une température plus élevée pour éviter de rester bloqué dans des optima locaux, puis de réduire progressivement la température au cours des itérations, permettant ainsi à l'algorithme de converger finalement.

3. Stratification - LAI Jinxing

3.1 Réalisation : principe de l'algorithme recuit simulé

Après de la configuration de la salle d'examen, nous initialisons tout d'abord aléatoirement la salle d'examen, Ensuite nous calcule l'énergie total avant faire la boucle, dans chaque itération nous permutons deux étudiants et puis calculer la différence d'énergie entre avant la permutation et après la permutation. Si jamais la différence d'énergie ΔU est non positive, cela signifie que nous réalisons une effective permutation, logiquement, nous gardons cette permutation. Sinon, ΔU est positive, qui signifie que cette permutation augment le niveau de regroupé du système. Au lieu de supprimer cette permutation non désireuse (comme les autres moyens de chercher un locale optimum), nous lui affectons une possibilité $e^{-\Delta U}$ en fonction de $-\Delta U$ pour garder cette permutation. C'est pour cette raison, cet algorithme possède une possible de sortir le local optimum et de converger un global optimum.

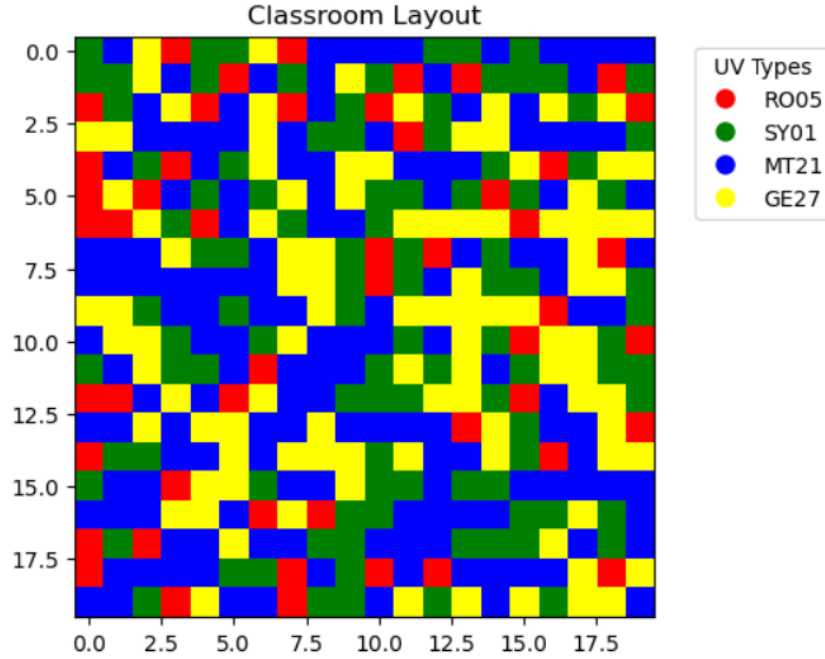


FIGURE 4 – Initialisation des places

3.2 Fonction d'énergie

La conception de fonction d'énergie est essentiel pour le réaliser en regroupé au lieu de séparé. Comme l'algorithme consiste à diminuer la valeur d'énergie, nous définissons la fonction d'énergie en façon de compter le nombre de la même UV environ chaque étudiant. S'il y a un étudiant avec même UV dans ses quatre directions, incrémenter un, sinon incrémenter 0.

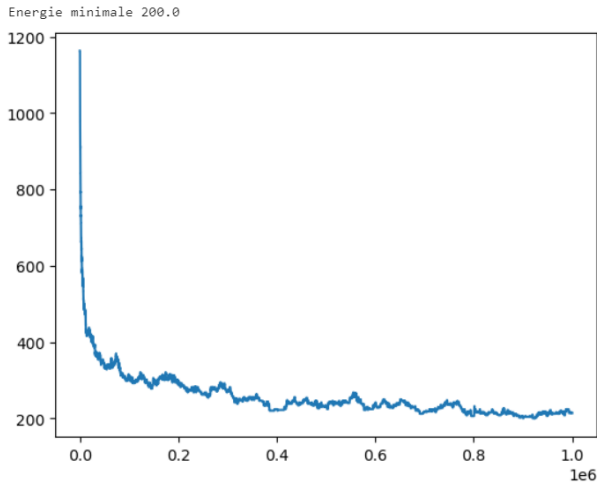


FIGURE 5 – Énergie en fonction du nombre d'itérations avec température égale à 1.0

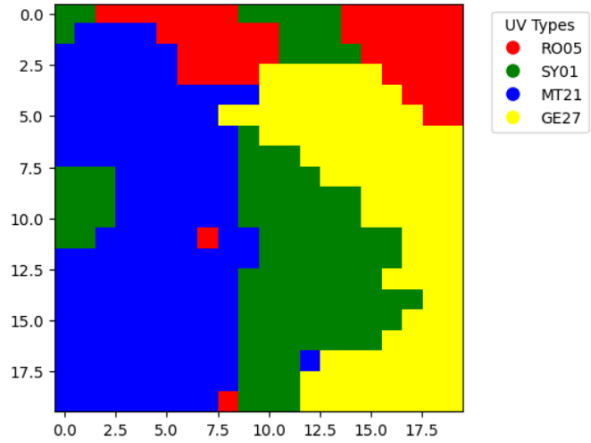


FIGURE 6 – Résultat obtenu avec température=1.0 et itération=10000

3.3 Conclusion

En conclusion, L'algorithme du recuit simulé nous permet non seulement de séparer des étudiants dans une salle d'examen, mais aussi de regrouper des étudiants, en outre nous permet de résoudre des variés problèmes d'optimisation en utilisant une effective fonction d'énergies et en modélisant bien le problème.

Voici le résultat obtenu suite à une simulation réalisée avec une température fixée à 1.0 et un nombre total de 716 100 itérations.

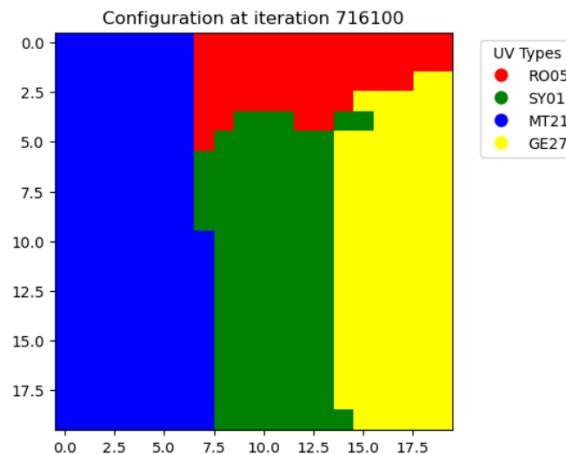


FIGURE 7 – Résultat obtenu avec température=1.0 et itération=716100