**OS202**

# Rapport projet python : recherche de chemin optimal

Baptiste MONTAGNES

8 mars 2024

# Table des matières

# Introduction

Ce projet Python est un projet qui utilise un algorithme d'optimisation par colonie de fourmis pour trouver le chemin le plus court dans un labyrinthe. Il crée d'abord le labyrinthe, puis l'affiche avec Pygame. Ensuite, il utilise on itère l'algorithme pour trouver le chemin optimal à travers le labyrinthe. Un nid de fourmis est placé à un coin du labyrinthe et de la nourriture est placée à l'autre bout du labyrinthe. Les fourmis explorent dans un premier temps le labyrinthe d'une manière aléatoire. Une fois que une fourmi a obtenu de la nourriture, elle revient sur ses pas et sécrète des phéromones aux endroits ou elle passe. Les fourmis prochent de dépots de phéromones sont attirées par ceux-ci.

# 1 Parallélisation simple sur 2 processus

Dans cette partie, l'objectif est de paralléliser le code sur deux processus. Un premier s'occupe de gérer l'affichage du labyrinthe, des fourmis et des phéromones avec la bibliothèque pygame de python. Le deuxième processus s'occupe de déplacer les fourmis dans le labyrinthe et de mettre à jour les phéromones.

Après cette parallélisation, on effectue des tests en arrêtant les itérations une fois qu'une certaine quantité de nourriture a été rapportée au nid de fourmis. On obtient les résultats suivants :

TABLE 1 – Speedup pour différentes tailles de labyrinthe et quantités de nourriture d'arrêt

| Taille labyrinthe | Quantité nourriture d'arrêt | Speedup |
|---|---|---|
| 10 | 200 | 1,265 |
| 10 | 1500 | 1,389 |
| 10 | 3000 | 1,172 |
| 20 | 200 | 1,213 |
| 20 | 1500 | 1,294 |
| 20 | 3000 | 1,127 |
| 25 | 200 | 1,357 |
| 25 | 1500 | 1,020 |
| 25 | 3000 | 1,226 |

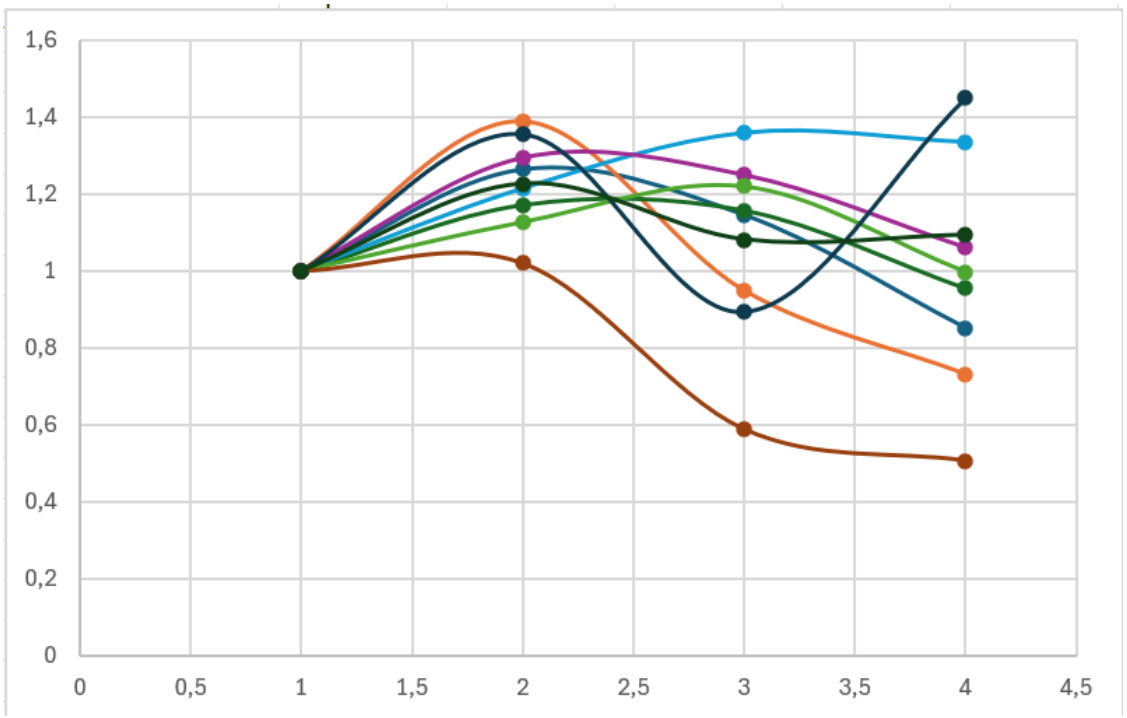
On remarque que le Speedup n'est pas très conséquent par rapport à la version non parallèlisée. On détecte que le Speedup est généralement plus élevé pour des tailles de labyrinthe plus petites et pour des quantités de nourriture d'arrêt plus faibles. En effet, plus la taille du labyrinthe augmente, plus la mise en commun des cartes de phéromones et la création de la nouvelle carte qui est réalisée sur un seul processus ralentit le code.

# 2 Parallélisation des fourmis sur plusieurs processus

On reprend le principe de parallélisation de la partie précédente sauf que cette fois ci, la gestion des fourmis et des phéromones est gérée par plusieurs processus. L'idée est de créer une colonie de fourmis par processus qui sera géré par ce processus là. Chaque processus va ensuite mettre à jour la carte des phéromones à partir des fourmis présentes dans sa colonie. Ensuite, l'ensemble des cartes de phéromones sont mises en commun sur un seul processus. Celui-ci va s'occuper créer la nouvelle carte de phéromones telle que pour chaque case, la

valeur de phéromones sera égale au maximum des valeurs de phéromones de la même case sur l'ensemble des cartes. Cette nouvelle carte est ensuite repartagée à tous les processus s'occupant de la gestion des fourmis.

Pour les mêmes variations de taille de labyrinthe et quantité de nourriture d'arrêt que à l'étape précédente, on obtient les Speedups suivants :




Les résultats ne sont pas satisfaisants. On obtient pour quelques configurations des Speedups inférieurs à 1. Cela peut être dû au surplus de communications nécessaires entre les processus pour le bon fonctionnement du code. Nous avons toutefois un résultat plutôt correct pour la configuration avec la taille du labyrinthe égale à 20 et une quantité de nourriture d'arrêt égale à 1500.

# 3 Réflexion sur la partition du labyrinthe entre les processus

On peut réfléchir à partitionner le labyrinthe entre les différents processus. Une idée serait que on divise le processus de manière géométrique en différents secteurs. Par exemple, pour 4 processus s'occupant de la partition du labyrinthe, on peut diviser le labyrinthe en 4 labyrinthes carrés plus petits. Chaque processus pourrait alors s'occuper des fourmis et des phéromones présents dans la zone gérée par le processus. Cette solution ne nécessite pas beaucoup de communication entre les différents processus et tout est quasiment parallélisé. Le problème est que la présence du nid et de la nourriture dans des coins du labyrinthe font que la répartition des fourmis et phéromones dans les sous-labyrinthes ne serait pas égale.

On peut alors penser à répartir de manière aléatoire les cases du labyrinthe entre les différents processus. Chaque processus devrait alors gérer plein de cases éparpillées dans le labyrinthe ainsi que les fourmis et les phéromones qui sont présents dessus. Le problème est que les fourmis vont beaucoup devoir changer de processus et que le procédé de communication serait beaucoup trop lourd et lent. Cette solution est donc elle aussi non adaptée.

Je ne vois donc pas d'idée de partition du labyrinthe qui serait efficace et optimisée. Ka partition des fourmis me paraît plus adaptée.