

ST7 Projet Intel

Colonies de fourmis - Soutenance mi-parcours

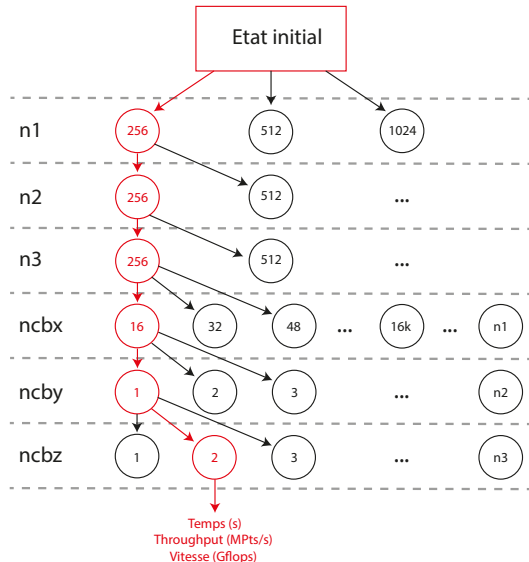
Simon Maréchal, Matthieu Oberon, Emile Prost
Carlos Santos García, Paul Saurou

24 mars 2021

1. Modélisation du graphe
2. Stratégies implémentées
3. Parallélisation
4. Difficultés rencontrées
5. Résultats avec 400 générations

Modélisation du problème

- Modélisation à travers d'un graphe
- Les tailles des caches sont limitées à la taille des matrices du problème
- Chaque fourmi est représentée par un 6-uplet et des valeurs de fitness obtenues
- Pour la suite, on se concentrera sur la vitesse de calcul.



Modélisation du problème

Matrice des phéromones

Il s'agit de la matrice d'adjacence de notre graphe qui permet de stocker les phéromones présentes sur chacune des arêtes. La taille de la matrice est fixe.

$$\tau = \begin{pmatrix} 0 & \tau_{1 \rightarrow n1} & & & & \\ & \tau_{n1 \rightarrow n2} & & & & \\ & & \tau_{n2 \rightarrow n3} & & & \\ & & & \tau_{n3 \rightarrow ncbx} & & \\ & & & & \tau_{ncbx \rightarrow ncby} & \\ & & & & & \tau_{ncby \rightarrow ncbz} \end{pmatrix} \in \mathbb{R}^{1098 \times 2122}$$

À la fin de chaque itération, τ est mise à jour.

À améliorer

L'implémentation de τ pourrait se faire à l'aide de matrices creuses afin d'optimiser l'utilisation de la mémoire

Stratégies implémentées

Voici les stratégies implémentées :

- Classic Ant System : Toutes les fourmis laissent des phéromones dans leur parcours du graphe
- Élitiste : La meilleure fourmi laisse deux fois plus de phéromones dans le graphe
- Max-min Ant System : Les phéromones dans le graphe restent entre deux valeurs

À faire

- Implémenter Ranked Ant System où les k meilleures fourmis laissent deux fois plus de phéromones sur le graphe
- Tester l'implémentation de "fourmis folles" qui, avec une probabilité de ϵ , explorent un chemin aléatoire.

Première implémentation de la parallélisation

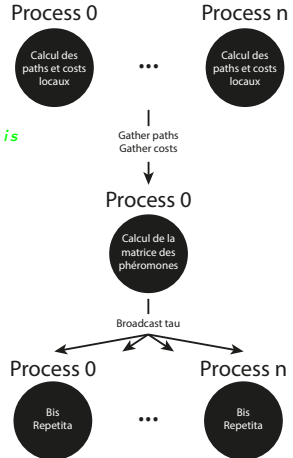
```
comm = MPI.COMM_WORLD
NbP = comm.Get_size()
Me = comm.Get_rank()

tau = tau0 # initialisation de la matrice de pheromones
for iter in range(n_iter):
    paths = []
    costs = []
    # calcul de chemins et leurs couts pour une sous famille de fourmis
    for ant in range(nb_ants/NbP):
        path = compute_path(tau, ...)
        paths.append(path)
        costs.append(compute_cost(tau, ...))

    # gather des chemins et couts des sous familles de fourmis
    comm.Gather(paths, all_paths, root=0)
    comm.Gather(costs, all_costs, root=0)

    # calcul de tau et du meilleur chemin et son cout
    if Me == 0 :
        best_path = ...
        best_cost = ...
        tau = compute_tau(tau, all_paths, all_costs, ...)
    # broadcast de tau
    comm.Bcast(tau, root=0)
```

PARALLELISATION PAR BROADCAST



Parallélisation finale

```
comm = MPI.COMM_WORLD
NbP = comm.Get_size()
Me = comm.Get_rank()

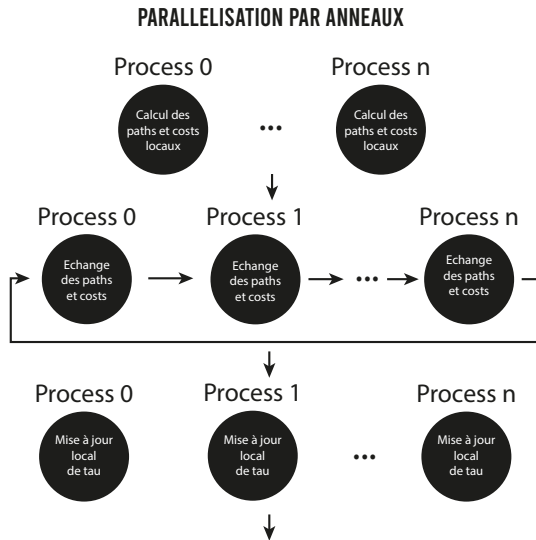
tau = tau0 # initialisation de la matrice de pheromones
for iter in range(n_iter):
    paths = []
    costs = []
    # calcul de chemins et leurs co ts pour une sous famille de fourmis
    for ant in range(nb_ants/NbP):
        path = compute_path(tau, ...)
        paths.append(path)
        costs.append(compute_cost(tau, ...))

    # on communique les chemins et co ts des sous-familles de fourmis dans l'anneau
    for i in range(1, NbP):
        comm.Sendrecv_replace(paths, dest=(Me+1)%NbP, source=(Me-1)%NbP)
        comm.Sendrecv_replace(costs, dest=(Me+1)%NbP, source=(Me-1)%NbP)
        tau = compute_tau(tau, paths, costs, ...)
```

Parallélisation finale

Matrice de phéromones τ initialisée sur tous les processus. Tant que le nombre maximal de générations de fourmis n'est pas atteint :

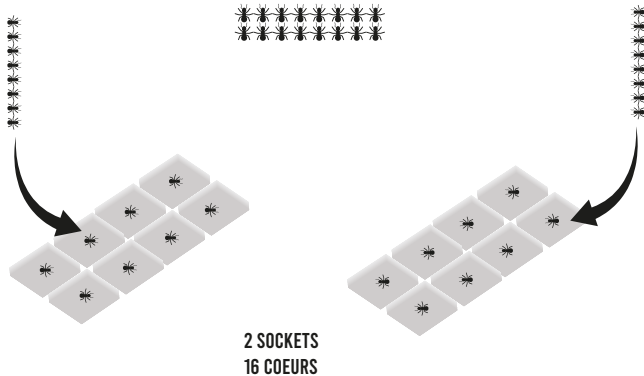
- Chaque processus calcule les **paths** et **costs** des fourmis associées
- Une fois une génération terminée, les processus communiquent en anneau et mettent τ à jour.



Et on recommence autant de fois qu'on a de process

Premier déploiement : Ants bind to core

SYSTEME DE COLONIE DE FOURMIS

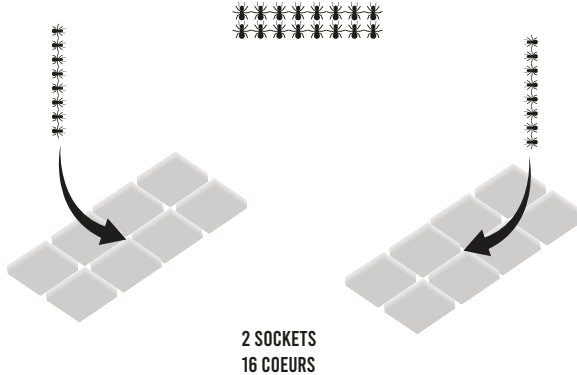


Problème

On ne profite pas du multithreading de iso3dfd !

Choix final : Ants bind to socket

SYSTEME DE COLONIE DE FOURMIS



Solution choisie

Cette solution permet à iso3dfd de créer un thread par core.

Difficultés rencontrées

- Prise en main du cluster
- Paramètre (nombre de threads) codé en dur dans le programme d'Intel
- Restructuration du code pour la prise en compte des paramètres $n1$, $n2$, $n3$
- Difficultés au moment de lancer des batches

```
ssh cpust75_8@phome.metz.supelec.fr
cpust75_8@kyle68:~/st7-intel/Appli-iso3dfd$ bin/iso3dfd_dev13_cpu_avx512.exe
256 512 256 8 100 256 27 17 n1=256 n2=512 n3=256 nreps=
100 num_threads=8 HALF_LENGTH=8
n1_thrd_block=256 n2_thrd_block=27 n3_thrd_block=17
allocating prev, next and vel: total 384 Mbytes
-----
time:          23.36 sec
throughput:    122.28 MPoints/s
flops:        7.46 GFlops
cpust75_8@kyle68:~/st7-intel/Appli-iso3dfd$ mpirun -np 1 -map-by ppr:1:socket
-bind-to socket bin/iso3dfd_dev13_cpu_avx512.exe 256 512 256 8 100 256 27 17
n1=256 n2=512 n3=256 nreps=100 num_threads=8 HALF_LENGTH=8
n1_thrd_block=256 n2_thrd_block=27 n3_thrd_block=17
allocating prev, next and vel: total 384 Mbytes
-----
time:          3.00 sec
throughput:    953.75 MPoints/s
flops:        58.18 GFlops
cpust75_8@kyle68:~/st7-intel/Appli-iso3dfd$ 
[ slurm1 ] 0 bash (1 launch) 2 bash [ 03-24 11:09 ]
```

Résultats

```
Path followed at iteration 389 on process 0 by ant 1 : [256, 512, 256, 256, 27, 17] with cost equal to -994.24.  
Path followed at iteration 390 on process 0 by ant 0 : [256, 512, 256, 256, 27, 17] with cost equal to -995.83.  
Path followed at iteration 390 on process 0 by ant 1 : [256, 512, 256, 256, 27, 17] with cost equal to -992.88.  
Path followed at iteration 391 on process 0 by ant 0 : [256, 512, 256, 256, 27, 17] with cost equal to -995.28.  
Path followed at iteration 391 on process 0 by ant 1 : [256, 512, 256, 256, 27, 17] with cost equal to -989.53.  
Path followed at iteration 392 on process 0 by ant 0 : [256, 512, 256, 256, 27, 17] with cost equal to -993.65.  
Path followed at iteration 392 on process 0 by ant 1 : [256, 512, 256, 256, 27, 17] with cost equal to -987.91.  
Path followed at iteration 393 on process 0 by ant 0 : [256, 512, 256, 256, 27, 17] with cost equal to -991.7.  
Path followed at iteration 393 on process 0 by ant 1 : [256, 512, 256, 256, 27, 17] with cost equal to -997.1.  
Path followed at iteration 394 on process 0 by ant 0 : [256, 512, 256, 256, 27, 17] with cost equal to -990.65.  
Path followed at iteration 394 on process 0 by ant 1 : [256, 512, 256, 256, 27, 17] with cost equal to -994.65.  
Path followed at iteration 395 on process 0 by ant 0 : [256, 512, 256, 256, 27, 17] with cost equal to -994.76.  
Path followed at iteration 395 on process 0 by ant 1 : [256, 512, 256, 256, 27, 17] with cost equal to -989.64.  
Path followed at iteration 396 on process 0 by ant 0 : [256, 512, 512, 176, 17, 71] with cost equal to -846.63.  
Path followed at iteration 396 on process 0 by ant 1 : [256, 512, 256, 256, 27, 17] with cost equal to -987.86.  
Path followed at iteration 397 on process 0 by ant 0 : [256, 512, 256, 256, 27, 17] with cost equal to -991.93.  
Path followed at iteration 397 on process 0 by ant 1 : [256, 512, 256, 256, 27, 17] with cost equal to -991.79.  
Path followed at iteration 398 on process 0 by ant 0 : [256, 512, 256, 256, 27, 17] with cost equal to -988.4.  
Path followed at iteration 398 on process 0 by ant 1 : [256, 512, 256, 256, 27, 17] with cost equal to -992.16.  
Path followed at iteration 399 on process 0 by ant 0 : [256, 512, 256, 256, 27, 17] with cost equal to -993.9.  
Path followed at iteration 399 on process 0 by ant 1 : [256, 512, 256, 256, 27, 17] with cost equal to -987.65.  
Le chemin optimal est [256, 512, 256, 256, 27, 17].  
Le throughput associé est alors 999.46 MPoints/s.
```

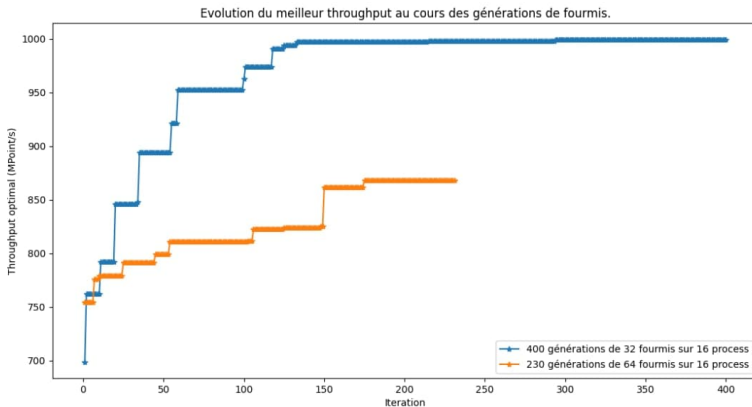
Remarque

On aboutit bien à une solution prise par une majorité de fourmis

Résultats

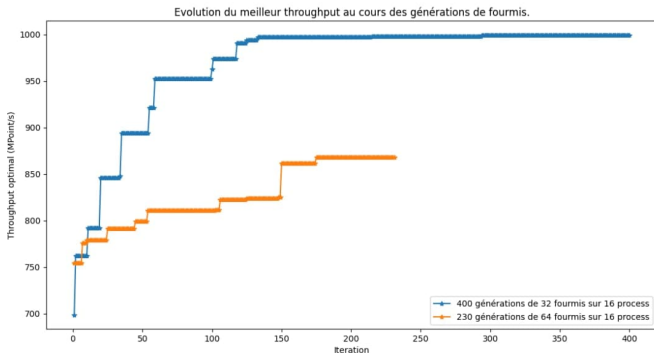
Expériences

On déploie 32 fourmis sur 16 process pendant 400 générations (en bleu) et 64 fourmis sur 16 process pendant 230 générations (en orange)



Analyse des résultats

- Coût en calcul identique pour courbes bleue et orange : les générations sont plus courtes pour la courbe bleue et les phéromones sont donc mises à jour plus souvent.
- Un grand nombre de fourmis prenant des chemins médiocres noie les quelques chemins optimaux pris dans la colonie



- Tester les autres stratégies de fourmis (pour l'instant seule la stratégie basique a été testée)
- Utiliser des matrices creuses
- Modifier les hyperparamètres pour converger plus rapidement
- Tester les paramètres de compilation
- Changer la fitness (consommation d'énergie ?)

Merci de votre attention
Questions ?