

ST7 Projet Intel

Colonies de fourmis - Soutenance finale

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

13 avril 2021

Aperçu

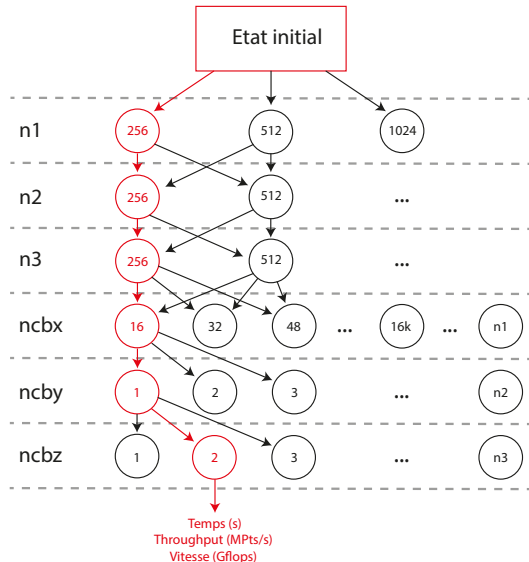
1. Modélisation du graphe
2. Stratégies implémentées
3. Parallélisation
4. Difficultés rencontrées
5. Résultats préliminaires
6. Stratégies : résultats obtenus
7. Conclusion

Sommaire

1. Modélisation du graphe
2. Stratégies implémentées
3. Parallélisation
4. Difficultés rencontrées
5. Résultats préliminaires
6. Stratégies : résultats obtenus
7. Conclusion

Modélisation du problème

- Modélisation à travers un graphe nécessaire pour ACO
- 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.

$$\tau = \begin{pmatrix} 0 & \underbrace{\tau_{1 \rightarrow n1}}_{1 \times 2} & & & & \\ 0 & & \underbrace{\tau_{n1 \rightarrow n2}}_{2 \times 2} & & 0 & \\ 0 & & & \underbrace{\tau_{n2 \rightarrow n3}}_{2 \times 2} & & \\ 0 & & & & \underbrace{\tau_{n3 \rightarrow ncbx}}_{2 \times 32} & \\ 0 & & 0 & & & \underbrace{\tau_{ncbx \rightarrow ncby}}_{32 \times 512} \\ 0 & & & & & & \underbrace{\tau_{ncby \rightarrow ncbz}}_{512 \times 512} \end{pmatrix} \in \mathbb{R}^{551 \times 1063}$$

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

Nos matrices denses sont remplies à 47%, donc l'utilisation de matrices creuses est non pertinente.

Sommaire

1. Modélisation du graphe
2. Stratégies implémentées
3. Parallélisation
4. Difficultés rencontrées
5. Résultats préliminaires
6. Stratégies : résultats obtenus
7. Conclusion

Stratégies de colonies de fourmis

Voici les stratégies implémentées :

- Classique : Toutes les fourmis laissent des phéromones dans leur parcours du graphe linéairement selon le gain de leur chemin

Stratégies de colonies de fourmis

Voici les stratégies implémentées :

- Classique : Toutes les fourmis laissent des phéromones dans leur parcours du graphe linéairement selon le gain de leur chemin
- Élitiste : La meilleure fourmi laisse deux fois plus de phéromones dans le graphe

Stratégies de colonies de fourmis

Voici les stratégies implémentées :

- Classique : Toutes les fourmis laissent des phéromones dans leur parcours du graphe linéairement selon le gain de leur chemin
- Élitiste : La meilleure fourmi laisse deux fois plus de phéromones dans le graphe
- Max-min : Les phéromones dans le graphe restent entre deux valeurs

Stratégies de colonies de fourmis

Voici les stratégies implémentées :

- Classique : Toutes les fourmis laissent des phéromones dans leur parcours du graphe linéairement selon le gain de leur chemin
- Élitiste : La meilleure fourmi laisse deux fois plus de phéromones dans le graphe
- Max-min : Les phéromones dans le graphe restent entre deux valeurs
- Fourmis ϵ -folles : Chaque fourmi choisit un chemin prometteur grâce aux phéromones déposées à l'itération précédente avec une probabilité de $1 - \epsilon$, et a une probabilité ϵ d'ignorer complètement les phéromones dans son choix de parcours pour partir explorer un autre chemin.

Stratégies de colonies de fourmis

Voici les stratégies implémentées :

- Classique : Toutes les fourmis laissent des phéromones dans leur parcours du graphe linéairement selon le gain de leur chemin
- Élitiste : La meilleure fourmi laisse deux fois plus de phéromones dans le graphe
- Max-min : Les phéromones dans le graphe restent entre deux valeurs
- Fourmis ϵ -folles : Chaque fourmi choisit un chemin prometteur grâce aux phéromones déposées à l'itération précédente avec une probabilité de $1 - \epsilon$, et a une probabilité ϵ d'ignorer complètement les phéromones dans son choix de parcours pour partir explorer un autre chemin.
- Différentiation des coûts : Les phéromones ne sont plus déposées linéairement selon le gain de chaque fourmi

Sommaire

1. Modélisation du graphe
2. Stratégies implémentées
- 3. Parallélisation**
4. Difficultés rencontrées
5. Résultats préliminaires
6. Stratégies : résultats obtenus
7. Conclusion

Algorithme à communication collective

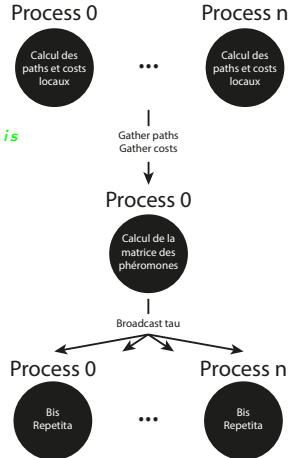
```
comm = MPI.COMM_WORLD
NbP = comm.Get_size()
Me = comm.Get_rank()

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



Algorithme à communication point-à-point

```
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, ...))

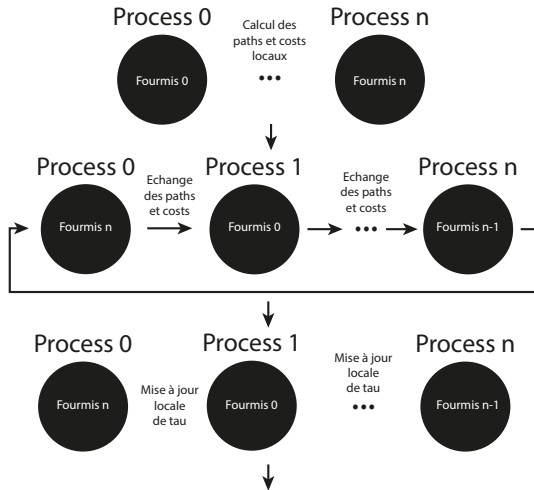
    # on met a jour tau avec ses propres donnees
    tau = compute_tau(tau, paths, costs, ...)
    # on communique les chemins et couts des sous-familles de fourmis dans l'anneau
    for i in range(1, NbP):
        comm.Sendrecv_replace(paths, dest=(Me+1)%NbP, source=(Me-1)%NbP)
        comm.Sendrecv_replace(costs,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

Parallélisation finale

La matrice de phéromones τ est 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.

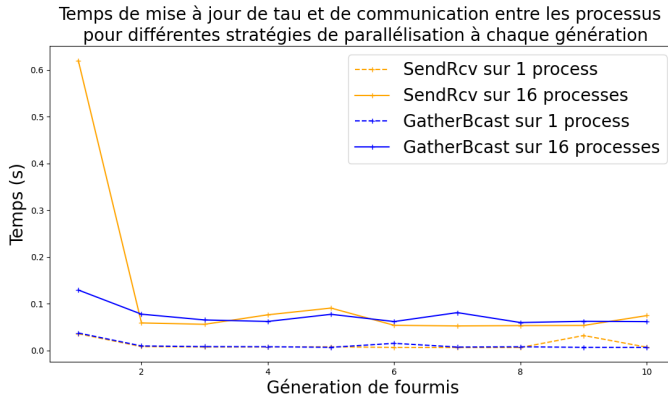
PARALLELISATION PAR ANNEAUX



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

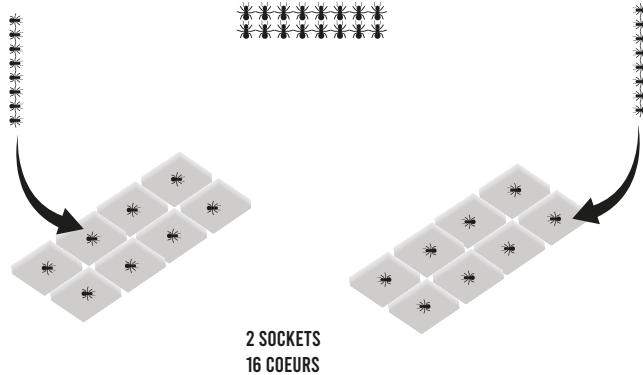
Temps de communication

Comparaison des temps de communication : on choisit la stratégie la plus adaptée au size-up.



Stratégie de déploiement : Ants bind to core

SYSTEME DE COLONIE DE FOURMIS

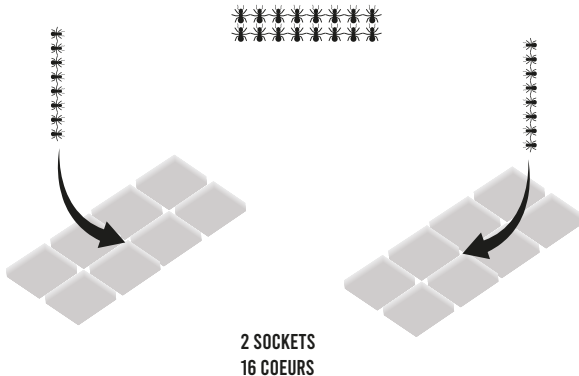


Problème

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.

Sommaire

1. Modélisation du graphe
2. Stratégies implémentées
3. Parallélisation
- 4. Difficultés rencontrées**
5. Résultats préliminaires
6. Stratégies : résultats obtenus
7. Conclusion

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 $n1$, $n2$, $n3$
- Appropriation de sbatch

```
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 ]
```

Difficultés rencontrées : multithreading

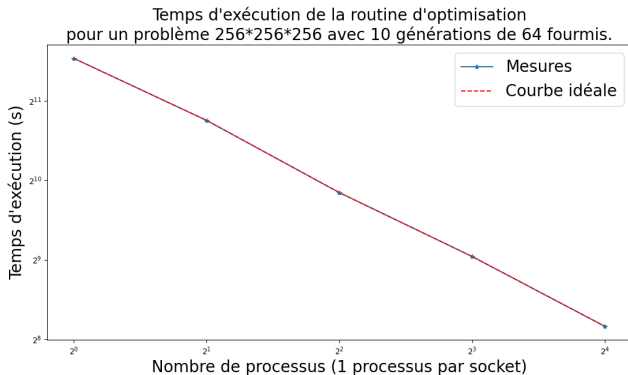
```
top - 16:31:24 up 18 days, 5:46, 1 user, load average: 4,48, 1,39, 0,50
Threads: 861 total, 9 en cours, 671 en veille, 0 arrêté, 0 zombie
%Cpu(s): 5,7 ut, 0,6 sy, 0,0 ni, 93,7 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem : 65420940 total, 46145280 libr, 13773284 util, 5502376 tamp/cache
KiB Éch: 33554428 total, 33554428 libr, 0 util. 50986328 dispo Mem
```

PID	UTIL.	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TEMPS+	COM.
20167	cpust75+	20	0	12,531g	0,012t	4328	R	25,1	19,2	0:20.58	iso3dfd_dev13_c
20170	cpust75+	20	0	12,531g	0,012t	4328	R	25,1	19,2	0:11.37	iso3dfd_dev13_c
20172	cpust75+	20	0	12,531g	0,012t	4328	R	25,1	19,2	0:11.37	iso3dfd_dev13_c
20173	cpust75+	20	0	12,531g	0,012t	4328	R	25,1	19,2	0:11.37	iso3dfd_dev13_c
20174	cpust75+	20	0	12,531g	0,012t	4328	R	25,1	19,2	0:11.37	iso3dfd_dev13_c
20171	cpust75+	20	0	12,531g	0,012t	4328	R	24,8	19,2	0:11.37	iso3dfd_dev13_c
20175	cpust75+	20	0	12,531g	0,012t	4328	R	24,8	19,2	0:11.37	iso3dfd_dev13_c
20176	cpust75+	20	0	12,531g	0,012t	4328	R	24,8	19,2	0:11.36	iso3dfd_dev13_c
20781	cpust75+	20	0	36016	4660	3188	R	1,0	0,0	0:00.32	top

Sommaire

1. Modélisation du graphe
2. Stratégies implémentées
3. Parallélisation
4. Difficultés rencontrées
5. Résultats préliminaires
6. Stratégies : résultats obtenus
7. Conclusion

Speed-up

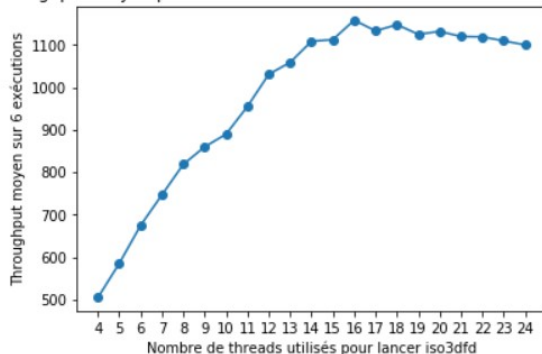


Analyse

Speed-up quasi-idéal pour une taille de problème fixée. L'initialisation et le temps de communication sont négligeables devant la partie parallélisable du code.

Multithreading

Throughput moyen pour l'exécution d'iso3dfd à nombre de threads variable



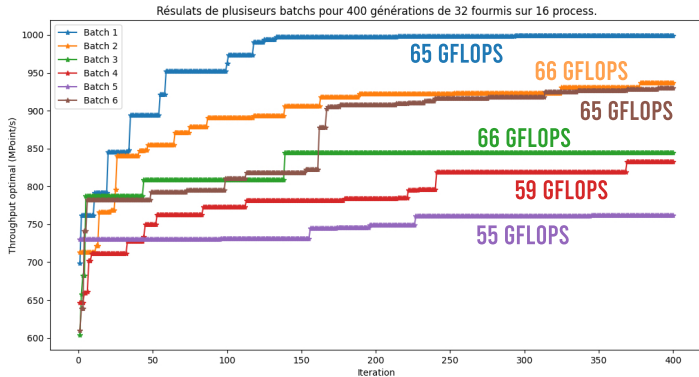
Analyse

On constate une amélioration du throughput jusqu'à 16 threads, qui correspondent aux 16 coeurs logiques par socket. Au-delà, les résultats se dégradent (hyperthreading)

Sommaire

1. Modélisation du graphe
2. Stratégies implémentées
3. Parallélisation
4. Difficultés rencontrées
5. Résultats préliminaires
6. Stratégies : résultats obtenus
7. Conclusion

Classic Ant Colony



Expériences

On déploie 32 fourmis sur 16 process pendant 400 générations. Les mesures sont faites avec iso3dfd lancé sur 8 threads.

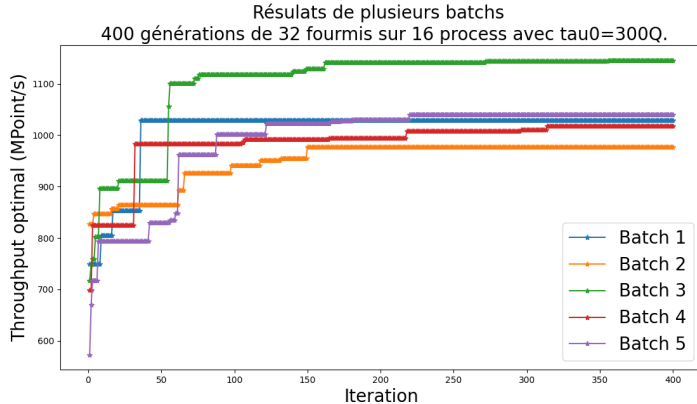
Classic Ant Colony : Chemins obtenus

Paramètres optimaux et throughputs associés retournés par les six batches

batch	n_1	n_2	n_3	cbx	cby	cbz	throughput ACO (MPoint/s)	throughput obtenu seul (MPoint/s)	GFlops obtenu seul
1	256	512	256	256	27	17	1000	1068	65
2	256	512	512	240	35	60	936	1062	66
3	256	512	512	224	7	28	844	1087	65
4	256	512	512	176	9	35	833	1074	66
5	256	512	512	80	10	10	762	974	59
6	512	512	256	256	25	9	930	895	55

Cohérent avec les résultats annoncés par Intel

Amélioration : Initialisation de τ

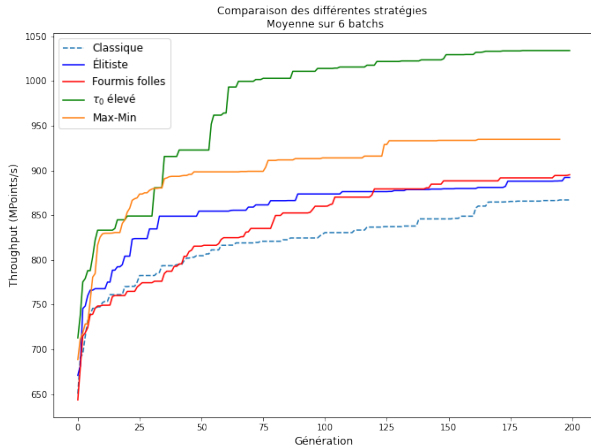


Analyse

On choisit une valeur initiale des phéromones plus importante pour permettre l'exploration de plus de chemins. Les résultats obtenus sont nettement meilleurs.

Comparaison des stratégies

- Les stratégies plus sophistiquées sont un peu meilleures
- L'initialisation a un impact important sur la performance de notre algorithme
- L'étude est à pousser car le temps de calcul sur le cluster est limité



Résultats finaux

Chemins optimaux par stratégie sur 300 itérations et lancés sur 8 threads

Les chemins ont été lancés individuellement via la commande mpirun et comparés aux résultats annoncés par ACO.

Stratégie	n_1	n_2	n_3	cbx	cby	cbz	throughput (MPoint/s)	ACO	throughput (MPoint/s)	seul	GFlops seul
Classique	256	512	256	256	27	17	999		1087		66
Élitiste	256	512	512	224	3	34	1019		1043		64
Folles	256	512	512	256	9	232	1138		1131		69
Max-Min	512	512	256	496	17	30	1024		1118		68

On remarque que, comme attendu, cbx est toujours proche de sa valeur maximale.

Mesures des performances

Process plus performants que d'autres

Path followed at iteration 299 on process 0 by ant 1 : [512, 512, 256, 496, 17, 30] with cost equal to -718.73
Path followed at iteration 299 on process 1 by ant 1 : [512, 512, 256, 496, 17, 30] with cost equal to -754.07
Path followed at iteration 299 on process 2 by ant 1 : [512, 512, 256, 496, 17, 30] with cost equal to -748.86
Path followed at iteration 299 on process 3 by ant 1 : [512, 512, 256, 496, 17, 30] with cost equal to -762.38
Path followed at iteration 299 on process 4 by ant 1 : [512, 512, 256, 496, 17, 30] with cost equal to -802.74
Path followed at iteration 299 on process 5 by ant 1 : [512, 512, 256, 496, 17, 30] with cost equal to -804.59
Path followed at iteration 299 on process 6 by ant 1 : [512, 512, 256, 496, 17, 30] with cost equal to -717.34
Path followed at iteration 299 on process 7 by ant 1 : [512, 512, 256, 496, 17, 30] with cost equal to -746.17
Path followed at iteration 299 on process 8 by ant 1 : [512, 512, 256, 496, 17, 30] with cost equal to -803.93
Path followed at iteration 299 on process 9 by ant 1 : [512, 512, 256, 496, 17, 30] with cost equal to -786.55
Path followed at iteration 299 on process 10 by ant 1 : [512, 512, 256, 496, 17, 30] with cost equal to -1011.32
Path followed at iteration 299 on process 11 by ant 1 : [512, 512, 256, 496, 17, 30] with cost equal to -710.92
Path followed at iteration 299 on process 12 by ant 1 : [512, 512, 256, 496, 17, 30] with cost equal to -744.75
Path followed at iteration 299 on process 13 by ant 1 : [512, 512, 256, 496, 17, 30] with cost equal to -773.93
Path followed at iteration 299 on process 14 by ant 1 : [512, 512, 256, 496, 17, 30] with cost equal to -804.47
Path followed at iteration 299 on process 15 by ant 1 : [512, 512, 256, 496, 17, 30] with cost equal to -828.93

Réalisation de 20 mesures sur un chemin optimal

Le chemin [256 512 512 256 9 232] a un throughput moyen de 1097 ± 60 MPoints/s, et 66 ± 5 GFlops. Cette variabilité rend plus difficile le choix du chemin optimal

Sommaire

1. Modélisation du graphe
2. Stratégies implémentées
3. Parallélisation
4. Difficultés rencontrées
5. Résultats préliminaires
6. Stratégies : résultats obtenus
7. Conclusion

- Résultats satisfaisants
- Structures optimales trouvées similaires en forme de "frite"
- Mise au point des hyperparamètres complexe
- Modélisation sous forme de graphe peu adaptée à l'utilisation d'algorithmes de fourmis :
élagage de l'arbre et utilisation d'heuristiques difficile.

Idées d'amélioration

- Si un chemin obtient exceptionnellement un très bon score, ne pas le prendre en compte.
- Augmentation de la taille des matrices
- Modifier les hyperparamètres pour converger plus rapidement
- Tester les paramètres de compilation
- Changer la fitness (consommation d'énergie ?)

Merci de votre attention
Questions ?