Voyageur de commerce

Ahmed Chokri, Adem Daami ,Ryma Bensalah, Maissa Farhani

Exercice 3 : Calcul de complexité et comparaison

Question 1 : Calculer et justifier la complexité :

1. Algorithme génétique :

```
Croiser(G g1,G g2){
      //résultat du croisement
      G r;
      //l'idée est tout simplement pour chaque variable de choisir
        aléatoirement la variable de g1 ou de g2.
      Pour chaque variable v de r
            Si un nombre aléatoire de 0 à 99 est inférieur à 50 alors
                  copier la variable correspondante à v de g1 dans v
            Sinon
                  copier la variable correspondante à v de g2 dans v
            Fin Si
      Fin Pour

      Retourner r;
}
```

```
SelectionNaturelle(G g[],N){
// On trie les éléments à la fitness la plus grande vers ceux elle est la plus
  petite.
      Trier g par les g[i] dans l'ordre décroissant.
      Retourner les N premiers éléments de g
}
```

1) Complexité

Un calcul de complexité de l'algorithme abstrait nous montre qu'elle est en O(n^2) ou n est le nombre de villes. Détail du calcul : O (Algo Gène) = O(n^2) + O (Muter) + O (Croiser) + O(Sélection Naturelle) = O (n^2) + O(n) + O (2n) + O(n) = O (n^2)

2. Algorithme Recuit simulé :

1. Algorithme

1- Prendre une solution initiale S

2- Tant que la température minimale n'est pas atteinte

2-1 Tant que l'équilibre n'est pas atteint

2-1-1 générer un voisin aléatoire S' de S

2-1-2 Calcul de Δ
Δ = coût(S')-coût(S)

2-1-3 si $\Delta < 0$ alors $S \equiv S'$

sinon $S \equiv S'$
avec la probabilite $exp(-\Delta/T)$

2-2 $T = \alpha T$

Recuit Simulé

Complexité = O(n²)

On un boucle while dans un autre boucle while avec n est le nombre de villes d'après l'algorithme précédent on peut conclure que notre complexité total est n² après la négligence de complexité de condition if

3. Algorithme Recherche tabou :

Schéma de l'algorithme tabou de base

- Engendrer une configuration initiale s_0 ; s := s_0
- s* := s ; f* := f(s)
- T := {} // liste taboue
- Répéter
  m := le meilleur mouvement parmi les mouvements non tabous et mouvements tabous exceptionnels (améliorant la meilleure solution)
  s := s ⊕ m
  si f(s) < f(s*) alors s* := s ; f* := f(s)
  Mettre T à jour (ajouter cycliquement m à T) ;
- Jusqu'à <condition fin>
- Retourner s*

La complexité est o(n3) car on a une boucle for qui contient 2 boucles for imbriqué

4. Algorithme des colonies de fourmis

```
1  Début
2      Entrée m : nombre de fourmis par colonies ;
3      L : nombre de colonies ;
4      n : nombre de villes ;
5      N c ← 0 ;
6      D ij ← 0 // matrice Globale ;
7      d ij ←t0 // matrice local de phéromones ;
8      Initialiser listeTaboue // avec la ville de départ de chaque fourmi ;
9      // L'initialisation de la listeTaboue et de la matrice Dij ;
10     Placer chaque colonie au hasard dans un point (ville) de départ ;
11     // Construction parallèle des tours par les différentes colonies ;
12     pour K ← 1 à m faire
13         Construction d'une tournée par chaque fourmi de manière aléatoire ;
14         Dépôt progressif des phéromones dans la matrice dij de chaque colonie ;
15         Evaluation et choix de la meilleure colonie ;
16         Initialisation de la matrice Dij avec celle de la meilleure colonie ;
17     fin pour
18     // Construction de la solution optimale ;
19     Les colonies sont positionnées dans une ville de départ;
20     Tantque ( N_c < N_max ) Faire
21         pour i ← 1 à n faire
22             pour j ← 1 à L faire
23                 pour K ← 1 à m faire
24                     Sélectionner la ville V_i à être ajoutée ou tour en cour selon la
                       formule (1);
25                     Evaluer la solution de la fourmi K sur le trajet (i,j) ;
26                     Effectuer la mise à jour globale de la trace selon la paire de
                       ville (i, j) si elle est meilleure que celle des fourmis
                       précédentes selon la formule 5.2 et 5.3(evaporation) ;
27                     Insérer au fur et à mesure (i,j) dans liste tabou de façon à
                       construire progressivement la solution de K ;
28                 fin pour
29             fin pour
30         fin pour
31     Fintantque
32     S ← la meilleure solution : Chaque colonie fournit une solution partielle ;
33 Fin
```

Initialisation :(étape 1)

1. Les m fourmis sont réparties aléatoirement sur les n villes.
2. Pour chaque fourmi, la liste qui modélise sa mémoire contient sa ville de départ.

Voyageur de commerce

Ahmed Chokri, Adem Daami ,Ryma Bensalah, Maissa Farhani

Étant donné que l'on a supposé une interconnexion totale entre les villes, nous avons une complexité de **O (n² + m. L)**

Constructions du trajet, et dépôts progressif de phéromones plus évaluation des solutions:(étape 2)

les opérations de calcul de la ville suivante nécessite un balayage de l'intégralité des villes
La complexité est **O (n² . m. L)**

Evaporation des phéromones: (étape 3)

La complexité est **O (n²)**

Evaluation des trajets après chaque transition (étape 4) :

pour les L colonies, on compare les tours de m fourmis de chaque colonies. Chaque tour a une longueur de n éléments => alors la complexité **O (n.m.L)**

conclusion :

Pour obtenir la complexité globale on va additionner la complexité de l'étape 1 -> **O (n² + m. L)** , au produit du nombre total de cycle (soit NCmax) par la complexité globale des étapes 2 à 4
alors on va obtenir **O (n² + m. L + NCmax . n² . m .L)**
comme O(n²) est négligeable devant O(Ncmax.n².m.l)
et O(m.L) est négligeable devant O(Ncmax.n².m.l)
D'où , la complexité finale est de la forme **O(NCmax . n². m .L)**

2.1 Algorithme des fourmis

`Ant_algo L n m` signifie qu'on a lancé `n` fourmis `m` fois

```
#Ant_algo L 50 50
temps : 7158.09 - distance : 19.824

#Ant_algo L 500 10;;
temps : 13349.29 - distance : 21.452

#Ant_algo L 500 100;;
temps : 108123.34 - distance : 21.452

#Ant_algo L 10 2000;;
temps : 50176.23 - distance : 15.683 ::
```

2.4 Algorithme de recuit simulé

Le recuit simulé est surtout utile pour finalisé une amélioration, Ici on utilise le recuit simulé pour essayer d'améliorer l'algorithme par colonies de fourmis (celui qui a donné 15.68).

`Algorecuit L Ti To k m` signifie qu'on a une temperature initiale `Ti`, une température critique `To` et qu'on lance l'algorithme `m` fois avec un facteur décroissance de température `k`.

```
# Algorecuit L 50. 1. 0.999 100;;
temps : 545.95sec - distance : 12.266
(*Amelioration de l'algorithme génétique le
plus performant obtenu ci-contre*)

# Algorecuit L 50. 0.1 0.999999 1000;;
temps : 2304.89 sec - distance : 11,933
(*Amelioration de l'algorithme par colonie de
fourmis le plus performant obtenu ci-contre*)
```

2.3 Algorithme génétique

Il est très long et l'amélioration se fait très lentement après une heure (c.f. les tracés de stats en annexe).

`Algogenetique L n k m` signifie qu'on a lancé `n` generation `m` fois avec un facteur mutagène de `k`.

```
#Algogenetique L 20 2 500000;;
temps : 51790.711 sec - distance : 25.486

#Algogenetique L 20 2 1000000;;
temps : 109206.092 sec - distance : 24.479

#Algogenetique L 20000 10 1000;;
temps :  1082.185 sec - distance : 28.518

#Algogenetique L 20000 10 1000;;
temps :  106546.239 sec - distance : 24.742
```

Voyageur de commerce

Ahmed Chokri, Adem Daami ,Ryma Bensalah, Maissa Farhani

On peut synthétiser ce qu'on peut estimer être les meilleurs algorithmes par le tableau suivant :

Algorithme	parcours	temps
Fourmis	15.7	13h
Algo génétique	24.48	30h
Recuit simulé	11.9	38mn
Recherche Tabou	7.3	27mn

D'après ce que précède on peut remarquer que l'algorithme de recherche Tabou est le meilleur algorithme au point de vue parcours et temps puisqu' il nous donne les meilleurs résultats 7.3 pour le parcours et concernant le temps 27mn.

les ressources :

- **https://www.i3s.unice.fr/~crescenz/publications/travaux_etude/colonies_fourmis-200605-rapport.pdf**
- **https://poweredtemplate.com/01412/0/index.html**
- **Problèmes d'optimisation combinatoires probabilistes (archives-ouvertes.fr)**
- **Radet-Souquet.pdf (unice.fr)**
- **Radet-Souquet.pdf (unice.fr)**
- **Résolution du Problème du Voyageur de Commerce Métaheuristique - PDF Free Download (docplayer.fr)**
- **cours4_8.pdf (eeisti.fr)**